9

AUSTRALIAN Personal Computer

STRINGY FLOPPY -UNRAVELLED

AUSTRALIA'S LEADING MICRO MAGAZINE JULY, 1980 $1.95

FIRST BITE OF THE APPLE III

¼ million people made their first computer purchase from us.

We are ComputerLand, the number one retailer of small computers in the world.

Why did so many people join the ComputerLand team? Because, like you, they wanted to buy their first computer from a specialist: from someone who cared about their satisfaction.

We at ComputerLand carry the widest selection of small computer equipment available in Australia.

Micro•computers, printers, V.D.U.'s, synthesisers and software, all at prices you can afford too.

Our courteous sales and service staff can configure a computer system to your specifications, and in many cases deliver it to you that same day.

So, don't limit your options. Come to ComputerLand today and join the number one team. We're expanding the way your world thinks.

Adelaide	—	131 PIRIE STREET PHONE 223 5083
Brisbane	—	127 CREEK STREET PHONE 221 9777
Melbourne	—	555 COLLINS STREET PHONE 62 5581
Perth	—	197 ST. GEORGES TERRACE PHONE 321 4671
Sydney	—	55 CLARENCE STREET PHONE 29 3753

ComputerLand

© CLA Sydney July 1980

CONTENTS

Volume 1 No. 3 July 1980

3 NEWSPRINT
This month's news and views.

5 YANKIE DOODLES
Dramatic word on the Apple III – from Tom Williams in California.

7 BENCHTEST
Stephen Withers reviews the Tandy TRS-80 Model II.

11 COMPUTER ANSWERS
More questions and answers from Sheridan Williams and his team of consultants.

13 ANIMISTICS
Neil Frude takes us into a future world where computers are cuddly; a special APC Interrupt.

18 M68000–MOTOROLA'S SWEET 16
Nicholas Jarmany reports on a forthcoming addition to the range of 16-bit super micros.

21 COMPUTER GAMES
Big trees and the Alpha-Beta algorithm. David Levy continues the series.

24 PLOTTING IN THREE DIMENSIONS
Malcolm Banthorpe offers a simple program for evolving 3D representations.

27 THE COMPLETE PASCAL
Sue Eisenbach & Chris Sadler continue their series with Part 3 – loops.

34 CHECKOUT
Thomas Murphy gives his personal impressions of the Exatron Stringy Floppy.

37 CHOOSING A SYSTEM
Guidelines by Dr Jon Patrick of Prahran C.A.E.

39 PRACTISING A LITTLE MICRO-CONTROL
Mike Dennis deciphers the Z80's control signals.

42 PROGRAMMING – THE SIMPLE APPROACH
An easy starter to the noble art – by Mervyn J. Axon.

47 USER GROUP INDEX
News of Melbourne's Commodore users group.

47 FAX
This month: the Z80 instruction set.

48 IN STORE
APC's directory of micros.

50 PROGRAMS
Another mixed bag of handy/fun listings.

55 BLUNDERS
Our very own page of penance . . . previous months' glitches, reversed.

56 NEXT MONTH'S PREVIEW/ ADVERTISERS INDEX
Buzzwords and Systems will return next month.

Typesetting
Pam Typesetting
Jane Hamnell
London Editorial Office:
Sportscene Publishers (PCW) Ltd.,
14 Rathbone Place,
London, W1P 1DE.
Tel: 01-637 7991/2/3
Telex: 8954139 A/B 'Bunch' G.
London
London Advertisement Manager
Stephen England
(01-631 1786)
Consultants
John Coll, Mike Dennis, Miriam Cosic, Michael James, David Hebditch, Sheridan Williams, Dr Adrian Stokes, Dr Stephen Castell.

Published by *Australian Microcomputer Journal*,
P.O. Box 115, Carlton,
Victoria, 3053.
Telephone: (03) 82 5783.
Telex: AA30333.
P.O. Box 250,
North Sydney, N.S.W. 2060.

Cover Illustration
Conny Jude

Distributed by
Gordon & Gotch (A'asia) Ltd.,
MELBOURNE, SYDNEY.

Copyright Notice
All material contained within *Australian Personal Computer* is protected by the Commonwealth Copyright Act 1968. No material may be reproduced in part or whole without written consent from the copyright holders.

APC welcomes all unsolicited material (written, photographic & illustrative) and, although no guarantee can be given as to its safe return, reasonable care and attention will be exercised.

Guidelines for Contributors
APC welcomes articles of interest. Don't be put off if your style of writing is "under developed" . . . true worth lies in the content, and shaping features comes naturally to us! Manuscripts should not exceed 3,000 words and authors are asked to use triple spaced lines with a wide left-hand margin; diagrams, listings and/or photographs should be included wherever possible. Please enclose a stamped, self-addressed envelope if you would like your article returned.

Because of the foregoing, it is necessary to add that the views expressed in articles we publish are not necessarily those of *Australian Personal Computer*. Overall, however, the magazine will try to represent a balanced, though independent viewpoint. Finally, before submitting an article, please check it through thoroughly for legibility and accuracy.

Subscription Rates:
Australia $24.00 for 12 issues.
New Zealand $30.00 for 12 issues (surface mail). Prices include postage and packing. Supplies to specialist shops can be arranged by negotiation direct with the publishers.

Australian Editor
Sean Howard
Editor
David Tebbutt
Deputy Editor
Peter Rodwell
Executive Editor
Bruce Sawford
Advertisement Manager
Marie Pirotta
Advertisement Assistant
Josephine McKenna
Production Manager
Michael Thomas

Ready for business.

CBM™ Business Computer System.

From the truly bewildering torrent of superlatives, claims, promises and come ons in the computer marketplace, how do you choose a microcomputer that's right for you?

Four things determine the suitability of a microcomputer: Hardware, Software, Service and Price.

At Hanimex we've got it all together with Commodore Business Machines:

- ☐ All components and equipment are designed and manufactured by Commodore ensuring total System compatability. Every CBM System is tested prior to installation to guarantee maximum reliability from the day you plug it in.
- ☐ An array of software programmes is available for business and other applications including General Ledger, Creditors, Debtors and Word Processing.
  However, if you require additional or specialised programmes these can be produced easily and inexpensively because of the advanced software tools built into CBM Systems.
- ☐ Hanimex and your Commodore Dealer have a serious commitment to service. Thanks to modular design and Self-Diagnostics, problems can be identified and remedied quickly and complete customer maintenance service is available.
- ☐ And when you finally get down to the price you will find that the Commodore Business System is more computer for less money. And a computer that will pay for itself faster.
- ☐ Along with Commodore, we at Hanimex are planning for the future needs of the business and professional worlds. Our main customer is the small businessman, but professional programmers, engineers and others are also finding our products invaluable.
- ☐ We are proud to offer this high quality product at an incredibly low price and CBM Systems are available for immediate delivery from the nationwide network of Commodore Dealers.

commodore

Distributed by
HANIMEX
Commodore Business Machines Division
Ph: (02) 938 0275

HAN18 80

Computer ~ works

June saw the opening of the community-based resource project in Crows Nest, NSW. It is to be a place where kids and students can get to know computers; and as a resource base for community awareness and use of computers.

It will be open after school, Tuesday to Friday, and at weekends. The following services will be included: micros for kids, a resource library, software libraries and exchanges for most micros, program development tools, and user oriented development projects in systems software, educational software, and networking. There will also be an information and resource distribution network to support microcomputers in NSW schools.

Students and teachers can obtain Computerworks resources for use in their schools and individuals can become supporting users for the trivial subscription of $20.

This really is an exciting project and, needless to say, being community based, the more support it gets the more valuable it becomes. So get out your chequebook if you're an individual (or do whatever you have to do if you're a non-individual)and direct enquiries or subscriptions to Computerworks, Box 3143, GPO Sydney, 2001.

Down south

It's good to see advice becoming available for non-professional computer users who, with a little help (biased or otherwise) from their friendly dealer and from the printed word (to which you can't direct questions), are trying to get some grasp of the area.

Dr Jon Patrick, of the Prahran CAE, will be available for anyone who has queries about his/her micro.

The Data Processing Dept. of Prahran CAE has been involved with microcomputers for nearly two years, and is now setting up a Centre for Microcomputer Applications. The department has a Z–1 Cromemco which is used in its academic programs. Its current configuration is 80K memory partitioned for two users and dual 5" disc drives. It will shortly be expanded with the addition of two 8" disc drives.

The Data Processing Department is part of the School of Business and, as well as running its own courses, is geared to supporting other departments, notably Accounting.

Because of this, the department is strongly user oriented and has purchased commercial software to enhance its support of other departments; and it is setting up a databank of benchmark programs and test results for the testing and comparison of micros.

Staff at the college will give advice to the general public on the use or acquisition of micros, and will act as consultants if requested.

Enquiries to Dr Patrick at the Department of Data Processing, Prahran CAE, 142 High Street, Prahran, Vic. 3181.

Micropros & Cons

MicroPro Design will start trading in the Commodore CBM and Pet microcomputers from the 1st of July. Their technical staff will be offering support in custom interfacing and applications software. They've already developed some interfaces for the Commodore IEEE488 Bus, which is manufactured locally.

Also new at MicroPro is the latest in the MicroCon series. The economical microprocessor/controller has been in production for two years and the latest is the MicroCon/OEM (MC1/OEM).

It is a 6" by 4.5" printed circuit board which uses the 6505 microprocessor, with a 1K byte monitor in EPROM and 1K bytes of Read/Write RAM with provision for the addition of an extra 1K of ROM or RAM.

Also provided are eight bit TTL level input and output ports, and a serial RS232 interface, for use with the monitor program or for general purpose communications. Enquiries to MicroPro Design, PO Box 153, Nth. Sydney, 2060.

Disc for a Dec

ADE has introduced a new double-sided, double-density DEC RX02–compatible flexible disc memory system. Called the DSD 470, it reads and writes on both sides of eight-inch diskettes with a formatted capacity of one megabyte per diskette, or two megabytes of on-line storage.

According to Bill Anderson, of ADE, the DSD 470 provides the user with LSI-11/23 multiple level interrupt support, conforming to DEC's announced standard interrupt structure for all future peripherals.

The 470 has onboard diagnostics – a series of microprogrammed user-selectable routines which verify proper operation of the system, debug to the chip level, provide detailed status reports and generally prevent you making a mess of things.

It's small – the chassis is just 5¼ inches – and (if the editor can be prevailed upon to print the picture large enough) it is interesting to see exactly how it's put together.

Contact Mr. H. Logie, the National Sales Manager, at Anderson Digital Equipment P/L, 1 Expo Court, Mt. Waverley, Vic. 3149; Tel: (03) 543 2077.

Telex talk

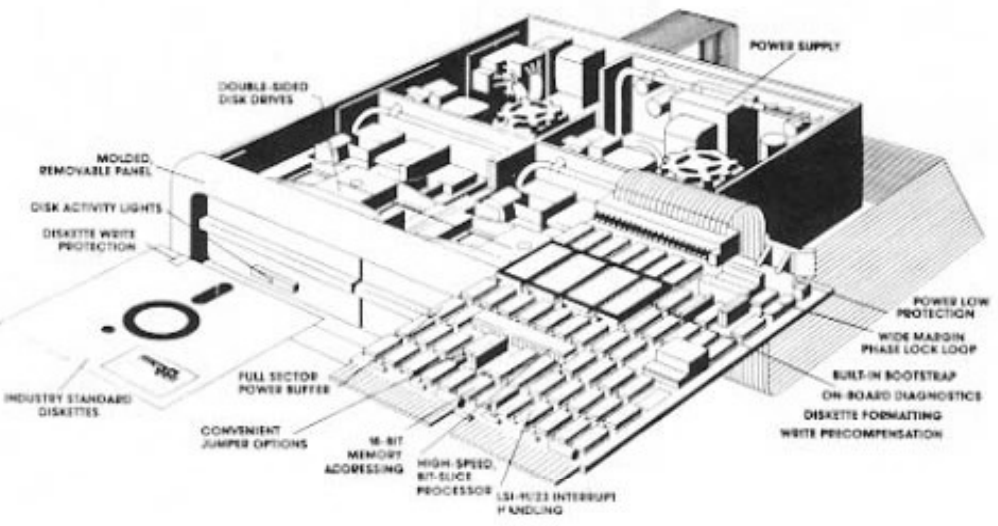
ADE has a new product combination, Teletaper and Telepunch, which prepares telex tape.

The message is prepared on the Teletaper's keyboard, displayed on a video screen, and then the unit's word processing capability can be used.

Once prepared the message is transferred electronically to the Telepunch. The tape is prepared off-line which frees the telex machine to send and receive more messages.

The Teletaper stores up to 32K of memory (16 pages), allowing recall of frequently used messages, while the word processing allows insertion of updated information.

It will cost around $5,000 and the operator apparently requires no telex training. Contact Anderson Digital Equipment P/L at P.O. Box 322, Mt. Waverley, Tel: (03) 543 2077 or P.O. Box 341, Pennant Hills, Tel: (02) 848 8533.



ADE's new DEC RX02 Disc Unit.

Hot off the press

A new book called *Microprocessor Software Design*, has been published overseas.

Edited by Max. J. Schindler, of *Electronic Design* magazine, it is a compilation of articles from that magazine and deals with application software from top-down design and operating system specifications to features of high-level languages such as Pascal and BASIC.

It hasn't been released yet, so enquiries to the Hayden Book Company, 50 Essex St., Rochelle Park, New Jersey, 07662; Tel: (203) 843 0550.

Pascalising Pet

The fact that APC is not written in French doesn't mean that French isn't quite a nice language (in its way). Rather it reflects the fact that most of the Frenchmen in Australia who want to read about computers can manage in English. The fact that there is going to be a version of the programming language Pascal for the Pet microcomputer does not prove that Pascal is much more than quite a nice language; instead it dictates that there are a lot of people who have learned Pascal, and find that the Pet's memory limit of 32K bytes prevents them from using their skill.

Getting a Pascal compiler into Pet isn't just a question of writing a program which will run on the Pet and translate Pascal instructions into strings of 6502 code. There's also the small problem of getting the program to fit inside 32K bytes, leaving enough room for it to write the 6502 code it generates.

The British micro-kit builder Transam has proved that its abilities lie beyond simple computer design by achieving that feat. That company has covered itself in further glory by winning the contract to get Pascal onto Pet – from the Pet's builder, Commodore, in the USA.

The new compiler, called TCL Pascal, sells for a comfortable £120 on diskette, complete with a large manual. They can be called, in London, on (01) 402 8137.

Super Pet

Despite Commodore UK's secretive attitude (the machine was launched at Hanover but the British press was banned from even seeing it until the official launch on the 14th of June), our London friends have managed to prise the following details of the new 80-column Pet from their reluctant grasp.

The green screen is a bumper 12 inches across instead of the usual nine, and the keyboard has been lowered to keep the overall height increase down to one inch.

As you can see from looking at the keyboard, the machine is totally business oriented: new features include repeat, tab, and ESC keys with auto repeat on the cursor controls. The 25 by 80 screen can be scrolled up or down and there are facilities for inserting and deleting lines and for defining "windows" on the screen. The machine, officially known as the 8032, has an 18K BASIC in ROM which includes disc operators.

Look out for a full Benchtest in APC soon.

The Sord is mightier than the pen

A lesser known computer in the field of business and personal computing is the Sord. Distributed by Mitsui & Co. in Australia, it is available from the Small Business Company in Sydney.

The M200 series are their micros. At the top of the line is the M223 MK V1 business system. This has 8.4 MB (formatted) Winchester Technology hard disc, single TEAC 350 K mini disc drive, CRT keyboard and 64K memory.

It is designed around the S100 bus and the back plane has four slots – one for the hard disc controller, one for the floppy disc controller, and two free for future expansion.

Built-in options include the keyboard which has ASCII characters for input, including graphic characters and a numeric keypad plus special function or option keys. These keys allow the operator to set up special repetitive functions, then leave the machine alone to do the work.

The CRT provides the ability of up to 1920 characters in an 80-character by 24-line display. Graphics include moving and reversible characters.

The built-in mini discs give 350K of user storage per disc. Two RS232 ports are switch controlled from the front panel. The ports can be configured independently for separate baud rates by two DIP switches located on the main board; and the B channel can be used with an acoustic coupler to communicate with another system.

The Sord operating system can control up to four disc drives – either as part of the built-in drives or as add ons. With the operating system comes Sord Extended BASIC, which also is available as a compiler version. Other languages are also available.

The ACE personal computer offers similar features to those of the M200 series but in a lower price range.

S.B.C. are themselves very interested in software development; and, apart from their own software development group, have set up contracts with outside software firms for the development of applications software for the Sord computer. puter.

Because they're now seriously going after the business market, S.B.C. are interested to hear from businessmen about their particular application needs. There are currently 35 software packages available ranging from Real Estate to Medical Practitioners, and come with documentation to guide the learner-user.

Sord systems are offered in a number of configurations, depending on needs. The prices vary according to system and size and type of printer and software. The basic M203 MKII Business System, with printer and software sells for $13,500.

Contact Small Business Computer Company at 200 Pacific Hwy, Crows Nest, 2065; Tel: (02) 929 7699.

Ohio level languages

In about a month you'll be able to get Fortran and Pascal for any disc-based Ohio system. It has already been released in the US by Ohio Scientific TCG. The minimum requirement is 48K of RAM, and they are expected to cost $600 in Australia.

Data 80

It all started in 1977 when the ACS and a couple of government bodies got Dataday together to provide average business people with information on their EDP requirements.

In 1979, Dataday became Dataweek; and this year it's being advertised to the general public. Data 80 includes a wide range of small business equipment exhibits, and a seminar series designed to provide the novice with basic information and the expert with a run down on latest developments.

Data 80 will be at Centrepoint in Sydney on August 5, 6, 7; and at the Oberoi Hotel in Adelaide on November 12, 13. Melbournians who haven't seen it already, have missed it!

Information from Graphics Directions Pty Ltd on Sydney (02) 212 4199. Ask for Miss Jana Pearce.

Hang in there

We've had enquiries from readers about the availability of Microsoft's BASIC compiler and the Apparat NewDos 80.

A couple of phone calls to the US revealed that some program bugs still need ironing out. So in spite of the extensive advertising, they're unavailable as yet. Shouldn't be long now . . .

Watch future issues for details of APC's tour to the 6th West Coast Computer Faire.

YANKEE DOODLES

Tom Williams, Editor-in-Chief of California's Info World, helps APC circumvent the Apple III information embargo by jetting over this up-to-the-minute report.

Myopia may not be an incurable condition, but it sometimes seems to require radical therapy. I give you the example of microcomputer manufacturers who are convinced that they have achieved the world's greatest hardware design. The fact that this design is different than anything else in existence is claimed to be one of the product's greatest assets, and well it may be from the standpoint of pure technological excellence. But when it comes to selling computers and providing the user community with products that are both useful and versatile, there are other considerations, considerations that require a little 'letting go' on the part of manufacturers.

It is a truism that the S100 bus is not the most refined design for micros. It's also true that, since S100 is not the proprietary design of any one company, there is more hardware and software available for it than for any other micro bus. This is not meant as a promotion for S100, but as an example of a phenomenon that was going on before everyone's eyes, and was misperceived by many to their detriment.

The example is that a major product which has been the result of much investment and design work can positively *benefit* from the existence of cottage industries. That seems so obvious that it's hardly worth saying. When Heathkit first announced the H-8 computer, they felt that a new cottage industry would spring up around their new Benton Harbor bus the way it had around the S100, simply because their new bus really was a design improvement over S100. That didn't happen, partly because there was not enough volume of Heath computers in the beginning also because the memory arrangements of the H-8 presented additional problems to software designers.

As a counter example, Radio Shack was able to make a success of the TRS-80 because they had a large volume of the machines available at startup, and because the TRS-80 was complete in that it required no expansion or configuration decisions on the part of the buyer before it could be used. Radio Shack has apparently resented the existence of independent manufacturers of peripherals for the TRS-80 and is rumoured to be designing custom-made chips into its new TRS-80/Color (or TRS-90) which will prevent the easy interfacing of non Radio-Shack devices to the machine. If that rumour is true, it's probably the biggest mistake Tandy could make.

Still more foolish are those companies who base the main software support for their machines on ROM cartridges. At first, this seemed like a novel approach. The first consumer computer to offer such a thing was the Video Brain by Umtech. You haven't heard much about the Video Brain of late, and there's a reason . . . it's no longer made.

It's one thing to provide the main system software in ROM - Exidy was the first to offer cartridges, but only for the language like BASIC or assembler — but it's quite something else to expect that all the applications programs will be provided in ROM packages as well. Not even the largest manufacturer can afford the human resources necessary to create the volume and variety of useful software demanded by users. And if there has been any lesson learned these past three years it is that *software* is what makes a computer valuable. Thus the only alternative is to make it easy for independent authors to write programs for the machine.

This cannot be done for ROM-based applications, because each author would need a development system for the computer in question, and that costs around $25,000. Texas Instruments and Atari have hedged a bit on this because they originally planned to have most applications programs on ROM. They've since come out with tape and diskette systems, but not the ROM. They seem unsure of their identity and have not attracted independent software vendors, and may be in trouble. The Texas Instruments machine is definitely in trouble and TI engineers who worked on the 99/4 project are said to be circulating their resumes because, currently, TI doesn't have anything in the works in the way of a personal computer.

There is evidence that the smarter companies are coming around to the realization that it's not only in their interest to allow ancillary entrepreneurs to produce both hardware and software products for their machines, it's also in their interest to *aid* them in doing so. When unveiling its new Apple III system, Apple said that it would be holding *seminars* for qualified independent hardware and software producers who wished to market products for the Apple III. Given this attitude and the very positive features of the Apple III, I predict Apple will have much success with this product.

Speaking of the Apple III, although as of the time of writing it had yet to be officially unveiled (its only airing prior to your reading this having been at the National Computer Conference in Anaheim). I recently got an advance peek at the machine and was quite impressed.

The Apple III has a CPU that's built around the 6502A with several other chips such that it executes a superset of the 6502 instructions. It also features relocatable base page register, relocatable stack, an and 128Kbyte address space. The basic machine comes with 96Kof RAM and is expandable to 128K.

The Apple III is supplied with a built in 5¼-inch disc drive, and 12-inch black -and-white monitor. Apple will be offering it as a complete 'problem solving' system. The first two such configurations to be offered will be a word processor and an 'information analyst'. The word processor will come with a second disc drive, a printer (there are several options), and word processing software. The information analyst will come with the single drive and Visicalc III, as well as a mail list manager and Apple business BASIC

The most impressive thing about the Apple III is the software orientation of its design. The display, which is now 80 characters by 24 rows, can be selected for any of 16 combinations of foreground and background colours. The character generator is in RAM, and is loaded when the operating system boots. This means that the entire set of 128 characters and symbols can be configured in software. A lookup table defines which letter, number, or symbol will be specified as each keyboard code comes in. Thus, any character set- Arabic, Greek, Japanese, Cyrillic, etc. — can be defined in software.

The software definable character set is also very useful in word processing operations. I saw some of the WP software under development and various type fonts were being displayed on the screen . . . medium, italic, boldface, etc. These, along with proportional spacing, corresponded exactly to what would appear on the printed page.

I/O is likewise very software oriented. Apple has written a large number of device drivers for most popular peripherals. When the system is configured, the user simply assigns a peripheral to a certain slot and assigns the proper device driver to it. Thus, whenever that device is called, the operating system takes care of slot and driver; the user simply says what peripheral to use.

Apple is also building in a battery-powered clock/calendar that it says will run continuously for three years. It's said to be accurate to one millisecond, and will keep track of year, day, month and time of day.

One other nemesis of Apple users has been cured: the reset button has been placed on the rear edge of the keyboard, and the control key must be pressed simultaneously to reset the machine. In addition, Apple has provided an Apple II emulation software package, which, when loaded into the Apple III makes it look exactly like an Apple II in terms of software and I/O. Thus, all the existing Apple II software can now be run on the Apple III.

Delivery of Apple III systems is scheduled to begin in late July or early August with the Information Analyst priced at about $4,400. The next will be the word processor around September, which will be priced at $5,400 to $7,800 depending on the type of printer chosen.

Technical specifications

CPU	2 MHz 6502-based with extended addressing
Memory	96-128Kbytes dynamic RAM, 4K ROM
Disks	1-4 minis, 143 kbytes/disc
Screen	Text: 40x24 b&w, 80x24 b&w, 40x24 in 16 colours, user definable characters; Graphics: 280x192 in six colours, 140x 192 in 16 colours, 560 x 192 b&w
Video	NTSC b&w or colour, RGB
Audio	Integral 2 inch speaker, six-bit DAC, one-bit square wave, 'beep'
I/O	RS232, two joysticks, printer output

DEFOREST SOFTWARE

GAMES

Sargon 2 — The ultimate Chess Game for any Microcomputer 16K Level 2. . . . . $37.50
Microchess — Level 1 or 2 TRS-80. Plays an entertaining game 3 levels fits in 4K. $24.00
Golf & Crossout — New improved version, good graphics. You can even design your own championship course. $10
Santa Paravia — up to 12 can play, become the king of a medevil city, levee taxes $10.00
Game Playing with Basic — 3 tapes 1, 2 & 3 based on the book of the same name great for learning . . . $12.50 ea.
Oil Tycoon — 2 players, explore, drill for oil, outprice the opposition, force him into bankruptcy. $10.00
Space Trek IV — population simulation, trade or wage war send missions to space, can you survive? $10.00
Checker King — not just an ordinary checker game but a fast expert machine language challenge. $27.00
Flight — Control your own aircraft, very realistic, good flying simulation and very challenging $10.00
Airmail Pilot — an entertaining game can you get the mail through in your 1927 Biplane, watch for lightening, windstorms, fuel shortages etc. $10.00
Adventures — what more can be said! The most challenging simulation/game/adventure ever Nos. 0 to 9. . ea. $16.00
1 on Disc $19.00
2 on Disc $32.00
3 on Disc $39.00

Backgammon/Keno — 2 great gambling games from instant software $10.00
Bandito — Your TRS-80 a slot machine with great sound effects $12.50
Star Trek — Acorn a space adventure on your TRS-80 . $12.50
Stock Market — Invest in safe stocks or high risk specs. will you take the risk $12.50
Original Adventure — requires 32K & disc from Microsoft. $35
Monolopy — Play that famous game against the TRS-80. . $10
Time Trek — with sound, another of the 'classics' for the TRS-80 real time excellent programme $21.00
Galactic Empire — a space game with a difference, many hours enjoyment with this one. $16.00
Light Pen — games & instructional program includes game frogs (Light Pen included) $10.00
Space Battles — requires 32K 1 disc $15.00
Games 20 — only 75 cents per program on disk $15.00
Pinball — the greatest game that I have ever seen for the TRS-80, fast action movable flippers and great sound. Machine language, maintains high score Disk. $20
Tape. $16
Super Space Invaders with Sound — nothing like the normal invaders extra fast. Spray bullets like out of a hosepipe, not only does the enemy move sideways but they also advance . . . . . .Disk $20
Tape $16
Lying Chimps with Sound — a very interesting program . . $10

COMPUTER BOOKS

Basic Basic $11.00
Advanced Basic $11.00
Basic from the Ground up $11.00
Basic Work-book $7.50
Discovering Basic $8.50
Common Basic Programs $12.50
Sargon Chess $19.00
How to Build a computer controlled Robot $15.00
How to profit from your personal computer $11.00
An Introduction to Microprocessors 0 . . . . . $11.75
" " " " 1 . . . . . $12.50
Z80 Programming for Logic Design $12.50
Z80 Assembly Language Programming $13.60
Pay Roll & Cost Accounting* *(Software to match $99)* . . . $20.00
Account Payable & Receivable* *(Software to match $99)* . $20.00
General Ledger* *(Software to match $99)* $20.00
6502 Assembly Language Programming $13.60
The 6800 Microprocessor $11.00
Basic Microprocessors & the 6800 . . . . . $16.00
Computer Mathematics $16.00
Consumers guide to personal computing & Microprocessors $11.00
Mini Computers, structure & programming $18.00
Fundamentals & Applications of Digitallogic circuits $13.00
Small Computer systems Handbook $11.50
The first book of Microcomputers $6.50
Computers in Society $9.50
Computers in Action $7.00
Standard Dictionary of Computers $25.00
Programming Programmable calculators $13.50
110 Basic Computer Programs $8.00
Cobol with style $9.50
Basic with style $8.00
Fortran with style $9.50
Pascal with style $8.50

Programming Proverbs $9.50
Fortran Fundamentals $6.50
Fortran 4 Programming $9.00
Microcomputer System Design $21.00
Microprocessor Data Manual $10.00
Microprocessor Basics $16.00
S-100 Handbook $10.00
Digital experiments $11.00
Digital Trouble Shooting $12.50
Telephone Accessories $7.50
More telephone accessories $7.50
Logical Design Using IC's $24.50
Statistical Pattern recognition $21.50
Fundamentals of Data Base Systems $21.00
400 Ideas for Design Vol 2 . . . . . $18.00
Vol 3 . . . . . $18.50
Vol 4 . . . . . $17.50
Integrity & recovery in Computer Systems $12.50
Management of Information Systems $11.00
File Structure for On-Line Core Systems $18.50
Data Management for On-Line Core Systems . . . $18.50
Digital Signal Analysis $27.00
Computer Security Risk Analysis $16.00
Character readers & Pattern recognition $17.50
Modern Electronics Security Systems $16.00
Printed Circuit Assembly $6.50
Basic Mathematics Vol 1 . . . . . $8.50
Vol 2 . . . . . $8.50
Mathematics for Electronics $9.50
Human Communication Handbook Vol 1 . . . . $12.00
Vol 2 . . . . $12.00
Computer Dynamics $13.00
Computer Aided Design Techniques $32.00
Electronic Game Projects $6.00
TRS80 Disk and other Mysteries "Pennington" . $25.00

bankcard
welcome here

Just ring with your Bankcard Number and Expiry Date and we will immediately forward goods.

POSTAGE IS INCLUDED AT NO CHARGE, HOWEVER WE SUGGEST REGISTRATION OR CERTIFICATION.

26 STATION STREET, NUNAWADING, VIC. AUSTRALIA
Tel: (03) 877 6946, 878 9276

BENCH TEST

TRS-80 MODEL II

The TRS-80 Model II is an attractively packaged integrated unit with a detachable keyboard and the sort of features that one would expect from a machine with a $5000 plus price tag. How has Tandy survived the move upmarket. . .? Stephen Withers (conducting his first Benchtest) reports.

The TRS-80 Model II . . . "functional and well designed".

The heart of the TRS-80 Model II system is a Z80A processor running at 4MHz. In order to minimise the load on the CPU, LSI controller chips are used to take care of the keyboard, video display, and disc drives.

The tested system contained 64K of memory and a single 8" Shugart disc drive. Up to three additional drives may be connected, each drive having a gross capacity of 497.25K (only 406.25K is available on Drive 0, as it contains the system disc). The Model II is also available with 32K of memory – if this option is selected it's possible to upgrade to 64K with one extra card. No other internal add-ons are available at present.

The Model II is not among the quietest of systems I've used. This is partly due to the cooling fan (I don't foresee overheating problems); the fact that the disc drive is permanently spinning also contributes to the noise. In addition this is likely to reduce the life of the media.

Two RS-232 ports are provided. These operate at the usual Baud rates, between 110 and 4800. Speed, word length, number of stop bits, and parity are all software selectable and this is much better than having to mess around with jumpers or switches. One channel may be operated in the less commonly used synchronous mode. A Centronics-compatible parallel printer interface is also standard.

The display shows a full 24 rows of 80 characters, and is clear and crisp. Contrast and brightness controls are fitted to the front of the casing, neatly hidden in the keyboard recess. The full set of printing ASCII characters are available (lower case characters have true descenders), and there's also a set of 32 rather strange graphics characters. I can't see these being used much, especially as they are not directly available from the keyboard. All characters may be shown in normal or inverse video.

The keyboard is connected to the main unit by a 2 foot long cable that terminates in a 5-pin DIN plug. Any spare cable tucks into the main cabinet. There are 76 keys – the normal QWERTY, plus a cursor control cluster, two "function" keys, and a number pad. Despite what you may have read

elsewhere, I believe a numeric pad to be very useful; it has been shown to reduce errors when entering numbers. A feature I appreciated was the indicator lights on the "caps" and "shift lock" keys.

The keyboard has most of the features one looks for: reasonably good "feel" (tactile feedback), n-key rollover, minimal reflection from the keytops, and a slim case that can easily be moved around the desk, yet is heavy enough to stay put while typing. One problem is that the "break" key is next to "backspace". Since "break" is used to halt a program, missing the "backspace" key can be very annoying! Mind you, thousands of Apple II users have learnt to live with a similar problem.

Software

When the Model II was first produced, it was supplied with a "prerelease" version of the system software, which seemed quite good, but had a number of weak points. In Australia the Model II uses a revised version. Having seen this new release, I'm glad to be able to report that almost every problem I noted has been fixed – and some new features added. Unfortunately this leaves me with less to write about!

Each time the Model II is reset it automatically executes an extensive diagnostic program which tests the disc sub-system, CPU, RAM, ROM, DMA controller and the input/output chips. If these tests are passed, TRSDOS is loaded and a large Tandy Corporation logo appears on the screen. At this stage the clock and data are initialised (unless the system has been PATCHed to skip this). If this feature is active the procedure is necessary after every reset. It could be argued that the action imposes a useful discipline on the operator, ensuring that all files are correctly dated.

Within seconds after switching on, the magic words "TRSDOS READY" appear, and you can get to work. All the usual functions of a disc operating system are present, so I'll only point out some of the more interesting or unusual features.

One of the first things you notice is that files may be protected by passwords – and not only that, there's provision for separate "access" and "update" words. For example, it may be necessary to give a clerk access to a file of sensitive information, but undesirable to allow him or her to be able to change or delete it. The level of access granted by the "access" word (from "no access" to "full, including KILL") is under the control of the holder of the "update" word. On top of this, each disc is given a password, knowledge of which allows the alteration of any user file's password and the deletion of any file.

Turnkey systems may be produced by utilising the AUTO command. If set, this causes the automatic execution of the specified command. This command would typically be "DO MYFILE", where MYFILE is the name of the file containing the TRSDOS commands which are to be executed. In this way it's possible to have the system automatically load BASIC and execute an applications program.

Although it's often useful to have a time function available, I found the clock display in the top right of the screen very distracting, and so left it switched off. The DATE command returns the time, date, day of the week, and the information that it is, for example, the 54th day of the year. Rather optimistically, Tandy say that this command will now work correctly beyond 2199 A.D. A utility which carries out date calculations (e.g. how many days between 3rd February 1979 and 28th October 1981?) is also supplied.

The DEBUG function is a simple machine code monitor. What makes it special is that it splits the display, reserving the top 13 lines for itself, while the remainder scrolls normally. While in "examine and alter memory" mode, it responds to the cursor control keys, allowing modifications to be made swiftly and easily, in a manner similar to the PET's screen editor.

Pressing ESCAPE makes the changes permanent, or the second function key cancels them. DEBUG will also accept input in Intel hex format through either RS-232 channel. I was rather surprised that a debugger is supplied without an assembler, as only the simplest programs are likely to be hand assembled.

In common with many of the commands, it's possible to send DEBUG's output to the printer, which may be a parallel or serial device. The printer driver is part of TRSDOS, and a function is provided to set its parameters (numbers of lines per page, etc) to suit the device.

Programs are provided to format and to copy discs. It's encouraging to see that FORMAT will test for bad sectors and mark any found as unusable, and that BACKUP uses all available memory as a buffer to speed the copying process and to minimise the number of disc exchanges necessary in a single drive system.

A utility that is likely to save much time and effort is PATCH. This allows the operator to modify a disc file (even a system file) by specifying its name, a target string and a replacement string. The main use is to correct bugs that may be discovered in the system software. It will only be necessary for Tandy to publish the two strings, which is far more convenient than recalling master discs for updating. PATCH is currently used to skip the initialisation of the date and time when booting up.

One last program that seems worthy of mention is the one that allows connection of the Model II to another computer as a fairly sophisticated terminal. The values of the control keys like "backspace" may be reassigned to suit the host system and information may be swapped between memory and disc.

Primitive functions like "get a character from the keyboard" are named Supervisor Calls, or SVC's (shades of the Jolly Giant!). These are documented and available to user-written programs by loading the accumulator with the appropriate SVC number, and then executing a RST 8 instruction. Other registers may be used to pass parameters. The zero flag is always set to indicate successful completion of the function. If it fails, an error code is generally returned in the accumulator. I counted 47 SVC's, most of them dealing with I/O functions, but some of them computational. One of the most interesting is "PARSER" which is used to split the contents of a text buffer into fields, with terminating and separating characters defined to suit the application. TRSDOS uses this function when processing a command line, separating the program or command name from the parameters, and the parameters from each other. The manual suggests PARSER would be useful as the kernel of a word processor – certainly an application to which the Model II would be suited.

Model II BASIC is licensed from Microsoft, and is said to be upwardly compatible with the TRS-80 Level 2 BASIC. Although it's the most comprehensive implementation I have used, I was surprised to find that it lacks matrix operations, WHILE statements, PEEK and POKE statements and multi-line function definitions. My main complaint is that variable names are limited to two characters, making intelligible programming difficult.

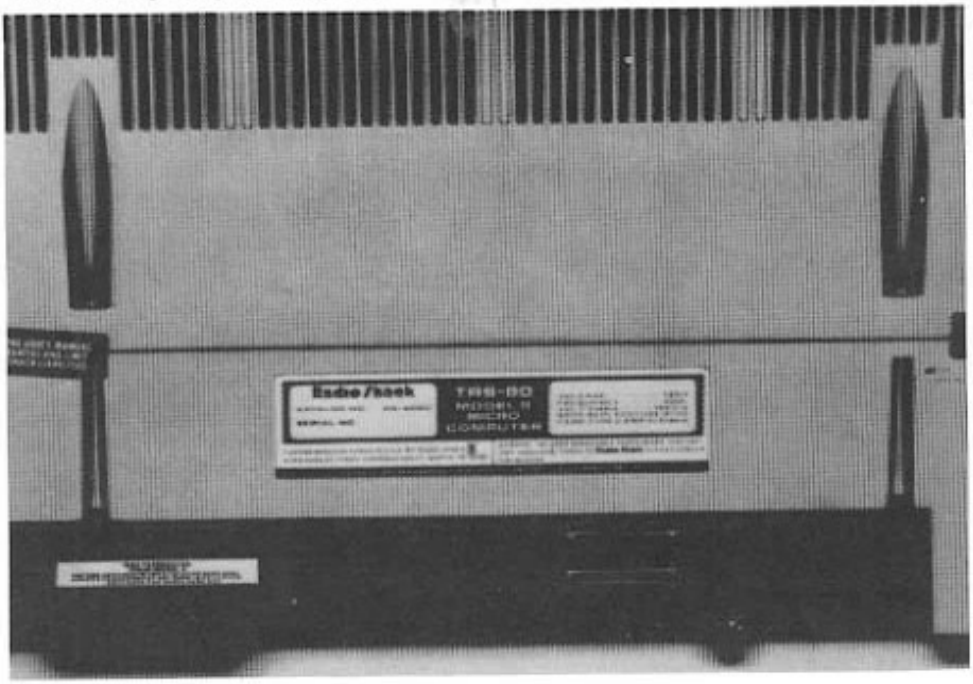
The price paid for all these features is the size of the program – TRSDOS and BASIC together occupy about 26K, plus an 834 byte buffer for each file used by BASIC. Clearly few users will be satisfied with a 32K system.

Real variables may be single or double precision, and are stored with up to 7 or 17 significant digits respectively. The type of a variable may be explicitly defined with a suffix (e.g. A% is an integer) or by using the DEFINT, DEFSNG, DEFDBL, or DEFSTR statements (e.g. DEFINT A). These are equivalent to FORTRAN IMPLICIT statements, where variable names beginning with the parameters of the statement default to the specified type. In the example all variables starting with the letter "A" would be integers, unless explicitly tagged with a suffix.

The interface between BASIC and machine code is well supported. In addition to 10 USR functions (whose entry points are defined within the program – no messing about with jump tables), there is a powerful function called VARPTR. This returns the address of numeric variables or the address of the pointer to string variables; it's possible to have machine code routines that operate directly on BASIC variables by passing their addresses as the parameter to the USR function. Machine code routines may be loaded into high memory by TRSDOS and then protected when entering BASIC by the use of the M parameter (which specifies the highest memory location available to BASIC).

Both random and sequential access disc files are supported. Using sequential files is straightforward, but I feel that random access (Tandy, incidentally, prefer the term "direct" access) has been made unnecessarily clumsy. After opening the file the program must define fields within the associated buffer; when a record has been read, variables are equated to these fields. To make matters worse, numeric values must be converted to strings before they can be stored on disc. Despite these criticisms, random access files work well on the Model II, and certainly they are easier to use than those on some other systems (Commodore, for example).

Above: The Model II keyboard has a lot going for it (see text) – only problem, the "break" key is next to "backspace".
Below: A tidy array of backplate.



The disc tests used are based on those developed by Sue Eisenbach; in fairness I should point out that the system was set to verify all disc writes (thus slowing things down), but on the other hand a virtually empty disc meant that the file was held in contiguous segments (which has the opposite effect). Strings in Model II BASIC have maximum length of 255 characters, so the tests involve a file of 100 records, each containing two fields of 128 characters.

Test 1 simply opens a new file and immediately closes it.

Test 2 uses a FOR loop to fill two strings (A$ and B$) with 128 "A"s, then opens an existing file; the second loop writes A$ and B$ into all 100 records, in ascending order. The file is then closed. Test 3 is similar, but the records are written in reverse order. This actually ran faster than Test 2 – Can anyone suggest why?

Test 4 opens the file, reads records 1 to 100, assigning the two fields to A$ and B$, and then closes the file. Test 5 repeats this process, but reverses the order in which the records are read. Test 5 was also faster than 4.

As an afterthought, I wrote a program which read 100 records selected at random. Although this involved a considerable amount of head movement, it was only fractionally slower than Test 4 or 5.

Potential

The Model II is unmistakably aimed at the business user. The full sized screen and good quality keyboard make it a natural for word processing. Since Lifeboat Associates supply CP/M configured for the Model II, as well as a good range of compatible software, this and many other applications are catered for "off the shelf".

Tandy offer a very limited selection of software for the Model II. I have seen their Mailing List package, which seemed to work well (detailed description would be unfair, as I didn't have a copy of the accompanying documentation).

Users who wish to stay with TRSDOS and BASIC have a very limited choice of software at present, although the file security aspects might make this option attractive. As so many other versions of BASIC are Microsoft products, it would not be excessively difficult to convert existing programs to run on the Model II in order to take advantage of its features.

The Model II is clearly one of the new breed of computers; powerful, integrated systems without some of the "sillies" that characterised an earlier generation.

I doubt that many of these computers will be sold for domestic or educational use, as the Model II's large disc capacity (probably its strongest point) is rarely an important factor in these environments.

Expansion

At present, expansion is limited to the addition of extra disc drives, and increasing a 32K system to 64K. As already stated, the motherboard allows for expansion when new devices become available (after all, Winchester discs are almost mandatory these days . . .). In case you feel that this lack of expansion is a bad point, when you have a 64K system with almost 2 megabytes of disc space, interfaces for printers, modems and what-have-you, as well as a full sized display and keyboard, what more do you want?

On the software side, Tandy are expecting the release of a Pascal system for the Model II in the near future, and at some stage, an assembler. I also heard that Fortran is in the pipeline, but as CP/M is available, who really cares? (Yes I know it isn't the world's best operating system, but it works,

Benchmarks

	Integer	Single Precision	Double Precision
BM1	1	1	–
BM2	4	5	6
BM3	13	13	41
BM4	13	13	43
BM5	14	14	44
BM6	20	23	52
BM7	30	35	65
BM8	6	6	7

DISC TESTS

TEST	TIME
1	3
2	39
3	38
4	20
5	19

(All times to the nearest second).

Prices Inc. sales tax

TRS80 Model II (32K)	$5,300
TRS80 Model II (64K)	$5,999
32K Memory Board	$699
1 Drive Disc	$1,999
2 Drive Expansion	$2,999
3 Drive Expansion	$3,999
System Desk	$599
Line Printer Stand	$199
Cable for Printer II	$69.95
Cable for Printer III	
10 Blank Discs	$99.95
Model II Manual	$39.95

and makes quality software available by the bucketful – that's enough for me!).

Documentation

All the documentation for the Model II comes in one three ring binder – which is nice, because it allows you to keep all your manuals together, even when you expand the system. (I only mention this because Tandy found it necessary to point it out in the manual!)

The description of the hardware (with setting-up instructions) is very brief, giving no information about the various peripheral controllers or other components. As the Model II is aimed at the business systems market, a detailed hardware manual is unlikely to be produced. However, there are substantial sections on TRSDOS and BASIC.

Although these two manuals were both produced on a dot-matrix printer, I am assured that they were draft copies and all systems will be supplied with properly typeset manuals. They are well laid out, giving an overview of the system before going onto a detailed description of the features. Each item starts on a fresh page, with its "syntax" and use described with the aid of one or more examples. Coupled with the index, this makes quick reference very easy. One exception is the "SYSTEM" command in BASIC. The manual points out that the TRSDOS "high overlays" may not be used through this command, but it doesn't list them, or even give a cross-reference to the appropriate page of the TRSDOS manual.

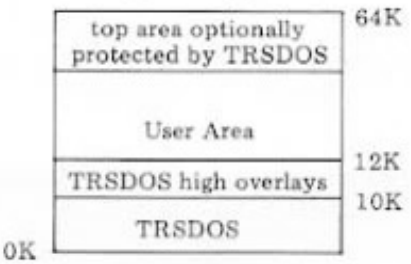
As far as quality is concerned, these manuals are as good as any I have seen. The only problem is that they are in the same style as those produced for mainframes – that is to say they are concise and definitive, but unsuitable for use as tutorial material. Indeed, the Model II makes this point explicitly, referring the reader to other books available from Tandy. Don't worry though, this only affects the programmer (who hopefully has some idea of what he/she is about); the machine itself is simple to operate. Given half-way decent software, it's well within the capabilities of the mythical "untrained typist". Parenthetically, all the typists I have met are far brighter than some advertisers' copy would have you expect.

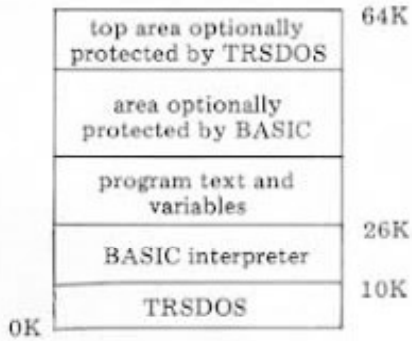
Conclusion

The TRS-80 Model II is an attractive, well designed computer. Its hardware incorporates all the features I expect to see on machines in this price range. Software, on the other hand, presents a dilemma – whether to stay with TRSDOS or to switch to CP/M. The first course severely (but temporarily) restricts the availability of applications programs, unless software from other systems is translated into the local dialect. Using CP/M avoids this difficulty, but sacrifices the excellent features of TRSDOS. Probably the fairest thing to say is that if I wanted a micro for a traditional data processing application, then the Model II would be on my shortlist.

TECHNICAL DATA	
CPU:	Z80A, 4MHz
Memory:	64K dynamic RAM, 1K "phantom" bootstrap PROM
Keyboard:	76 keys
Screen:	12" diagonal, 24 lines x 80 characters
Cassette:	N/A
Disc Drives:	One 8" double density floppy disc
Printer:	Not included in basic system
Bus:	Non-standard
Ports:	2 RS-232 serial, 1 Centronics compatible parallel
System Software:	TRSDOS
Language:	BASIC

Memory map





At a glance

FIRST IMPRESSIONS	
Looks	****
Setting-Up	*****
Ease of Use	****
HIGH LEVEL LANGUAGES	
BASIC	*****
FORTRAN	N/A
COBOL	N/A
PASCAL	N/A
System Software	****
PACKAGES	
Business	Available
Education	N/A
Home	N/A
PERFORMANCE	
Processor	****
Cassette	N/A
Discs	****
Peripherals	N/A
EXPANDABILITY	
Memory	***
Cassettes	N/A
Discs	****
Bus	**
COMPATIBILITY	
Hardware	**
Software	***
DOCUMENTATION	****
VALUE FOR MONEY	****

*****	excellent
****	v. good
***	good
**	fair
*	poor

RESERVED WORDS

Command Statements:

AUTO	DELETE	EDIT
KILL	LIST	LLIST
LOAD	MERGE	NEW
RENUM	RUN	SAVE
SYSTEM		

Program Statements:

Definition and initialisation

CLEAR	DATA	DEFDBL
DEFFN	DEFINT	DEFSNG
DEFSTR	DEFUSR	DIM
ERASE	RANDOM	REM
RESTORE		

Assignment:

LET	LSET	RSET
MID$=	READ	SWAP

Control:

END	IF..THEN..ELSE
GOTO	RETURN
ON..GOTO	GOSUB
FOR..NEXT	ON..GOSUB

Input/Output

INPUT	CLOSE
PRINT	INPUT #
PRINT USING	PRINT #
LPRINT USING	CLS
GET	PRINTTAB
OPEN	LPRINTTAB
LINEINPUT	FIELD
PRINT@	LINEINPUT #
LPRINT	PUT

Debugging/error trapping

CONT	ON ERROR GOTO
ERROR	STOP
RESUME NEXT	ERR
TROFF	RESUME
ERL	TRON

Functions:

ABS	ASC	ATN
CDBL	CINT	COS
CSNG	EXP	FIX
INSTR	INT	LEN
LOG	RND	SGN
SIN	SQR	TAN
VAL	CHR$	DATE$
HEX$	LEFT$	MID$
OCT$	RIGHT$	SPACE$
STR$	STRING$	TIME$
INKEY$	INPUT$	POS
ROW	SPC	CVD
CVI	CVS	EOF
LOC	LOF	MKD$
MKI$	MKS$	FRE
MEM	VARPTR	USRn

COMPUTER ANSWERS

Every month in APC, Sheridan Williams will assist readers with their hardware, software and systems difficulties. Some questions he will deal with himself, other enquiries will be directed towards members of his consultancy panel.

Trouble of a sort

I've written a package for my business and basically I'm very pleased with it. I use floppy discs to store customer records and frequently need to print these records in alphabetical order. I now have nearly 1000 records and hold these in customer number order. I use a method that sorts onto a second disc, which I can then keep, but the sort seems to take hours. Can you help with a faster routine?

P. Abbott.

You don't give a name to your sorting technique, but it's probably a version of the 'bubble' or 'ripple' sort. The majority of books and novice programmers use the 'bubble' sort, which must surely be the slowest sort ever invented. The reason for its popularity is probably twofold; it's very easy to understand, and only takes a few lines of coding. It has the annoying property that if we double the number of items being sorted, it will quadruple the sort time. On this basis if it takes one second to sort ten numbers, it will take four seconds for 20 numbers, and so on until it takes 16000 sec for 1200 numbers. Over four hours! Part of the problem lies in the fact that we are usually using interrupted BASIC which serves to slow down processing anyway.

What we require is a far more efficient sort. One that can be recommended for speed is called the 'quicksort' (clever, eh?). The quicksort only doubles the sort time for double the number of items being sorted. Using our previous example, if it takes one second to sort 10 numbers it will take two seconds for 20 numbers and 160 secs for 1200 numbers, which is just over 2.5 minutes. It would take too long to explain how the quicksort works but here is the coding. In this example it will sort N numbers in ascending order in the array A(1) —A(N). It requires the B array as working storage, but this array only needs 24 elements to sort 5000 numbers so there is not a lot of extra space needed:

```
10 DIM A(N), B(INT(LOG
(N)/LOG(2)+1),2)
1000 REM ** quicksort sub-
routine **
1010 S=1
1015 B(1,1)=1: B(1,2)=N
1020 L=B(S,1): R=B(S,2):
  S=S—1
1030 I=L: J=R: X=A(INT
  (RND(1)*(R—L)+0.5)
  +L)
1040 IF (A(I)>=X THEN
  1050 ELSE I=I+1: GOTO
  1040
1050 IF X>=A(J) THEN
  1060 ELSE J=J—1: GOTO
  1050
1060 IF I>J THEN 1080
1070 W=A(I): A(I)=A(J):
  A(J)=W: I=I+1: J=J—1
1080 IF I<=J THEN 1040
1090 IF J—L>=R—I THEN
  1140
1110 IF I>=R THEN 1130
1120 S=S+1: B(S,1)=I:
  B(S,2)=R
1130 R=J: GOTO 1170
1140 IF L>=J THEN 1160
1150 S=S+1: B(S,1)=L:
  B(S,2)=J
1160 L=I
1170 IF L<R THEN 1030
1180 IF S>0 THEN 1020
1200 RETURN
```

Having decided that this is the sort to use, let's look at the problem a little more deeply. If we have 1000 customer records each of length 100 ch, say, we would require over 100K of memory in which to sort the records. It's unrealistic to sort in memory because it's quite likely there will not be enough space. We will consider two alternatives: 1) sort entirely on the disc; 2) sort the record keys in memory and then access the records in that order. Method (2) will be faster as it takes far longer to swap records on the disc than it does to swap them in memory.

Method (2) will require that you read into an array the first four characters of the key field, followed by the record address. For example, if the disc file holds as record 1, WATERS & CO LTD etc, and record 2, BLOGGS MOTORS etc, then in the array D$(1) will hold WATE1 and A$(2) holds BLOG2. We must restrict the number of characters to a suitable figure to allow the total amount of records to fit into memory. That means, at eight characters per record we will only need around 10K of memory to hold 1000 records. For example to read record Y from disc and store it in array element X we would use:

```
READ/1@Y,T$: A$(X)=
  LEFT$(T$,4)+STR$(Y):
  X=X+1
```

Because BASIC is so different in various versions the statement READ/1@Y,T$ means read from file number 1 (disc file already defd maybe as OPEN/1,'CUST.DAT') at record number Y(direct access address) into string T$.

Once the array has been sorted the following program would print the file in sorted order:

```
100 DIM A$(1000)
110 OPEN/1, 'CUST.DAT'
120 FOR X=1 TO 1000
130 Y=VAL(MID$(A$(X),
  5))
140 READ/1@Y,T$
150 PRINT T$
160 NEXT X
170 CLOSE/1
```

Advantages of this method are that the file remains intact, and that sort is quick. Method (2) requires that the file itself will be sorted and hence it's best to sort a back-up copy of the file in case the system crashes in mid sort. This method has the advantage that a file of any size can be sorted, and that the sorted file can be kept as a permanent file if needed. The program is very similar to that above and the changes are as follows:

```
1030 I=L: J=R: READ/1@
  (INT(RND(1)*(R—L)+
  0.5)+L),X$
1040 READ/1@I,Y$: IF Y$<
  X$ THEN I=I+1: GOTO
  1040
1050 READ/1@J,Z$: IF
  X$<Z$ THEN J=J—1:
  GOTO 1050
1070 WRITE/1@I, Z$:
  WRITE/1@J,Y$: I=I+1:
  J=J—1
```

I hope these hints may help people who are struggling with sorts. If the above programs do not run then it's because of errors in transcription, please write if you have any difficulties.

S.W.

After BASIC?

Although I program in BASIC fairly well, I would like to consider another language that is more powerful than BASIC. Can you suggest any that are worthwhile trying, and would I find them difficult.

As you have written to PCW I will assume that you only want languages that are available on microcomputers. This does restrict you considerably, but it may not be a bad thing because only the most common languages have been implemented on micros so far. The main problem with implementing languages other than interpreted BASIC on micros is that as the majority are *compiled* they require a disc system to give of their best. They also require more than 16K in order to support the compiler and operating system. I have assumed that you are only interested in high-level languages not assembly languages.

FORTRAN — is the most widely used language in the scientific field. There are a great many programs already written in FORTRAN and hence this library should be available to you, saving you a great deal of programming time. FORTRAN has well defined input/output routines, and is universally defined, enabling programs to be as portable as possible. FORTRAN is, however, not a structured language, and this in many people's eyes it its main failing. FORTRAN has many niggling limitations that make programming tedious — for example DO loops (The FOR loop equivalent) will only work for integers greater than zero.

ALGOL — is the other main scientific language with a first class structured approach; algorithms are written using ALGOL-like statements, making translating into ALGOL particularly easy. Library routines are readily available, and techniques such as 'recursion' are possible. ALGOL's main failing is the lack of defined input/output routines. Both ALGOL and FORTRAN have very limited string handling routines.

COBOL — is a business language which needs a fairly large amount of storage. Programming in COBOL takes some time to master, but as this language is the world's most popular the rewards for learning it are worthwhile. COBOL is an English-like language using words rather than symbols — example: HTCMS=2.54 * HTINS would be MULTIPLY HTINS BY 2.54 GIVING HTCMS. Note that COBOL is not particularly appropriate to scientific applications.

PASCAL — is a recent attempt to marry all the advantages of other languages and remove all their restrictions. It is structured like ALGOL. As long as PASCAL is defined to a universal standard then it is probably one of the best languages to learn. It promises to be available on most micros eventually. Read the articles in previous PCWs for more detail on PASCAL.

FORTH — is available on several systems (sometimes in a version called FIFTH). FORTH is a 'threaded' language ideally suited to microcomputers as it only requires around 5-6K for the interactive FORTH compiler. FORTH requires no extra area for symbol tables, overlays or any other software. FORTH is very fast, certainly faster than any of the above languages, and allows assembler inserts if it is still not fast enough for your application. FORTH is ideal for compiler writing as well as 'ordinary' programs. All routines in FORTH operate using a stack and every time a new 'primitive' (key word) is defined it can be incorporated permanently in the language.

Sheridan Williams

SUBSCRIPTIONS

I would like to subscribe to Australian Personal Computer for one year (12 issues)

beginning with the month of .....

My Name .....

(Block letters please)

My Company
(if applicable) .....

My Address .....

.....

Date .....

Signature .....

☐ Australia $24.00

☐ Elsewhere A$30.00

☐ I enclose my cheque, made payable to Australian Personal Computer for $ .....

☐ Please invoice my company

Please send this entire order form, together with your remittance to Australian Personal Computer, Subscriptions Dept., P.O. Box 115, Carlton, 3053, Australia.

AUSTRALIAN Personal Computer

APC is the pre-eminent publication in the field of microcomputers. Whatever your area of interest in microcomputing – be it business, home, educational – or simply as a leisure pursuit – APC is your passport to the future. Subscribe today, and sleep soundly at night, safe in the knowledge that your personal copy of APC will come thudding through your letterbox every month.

Have a peek....

Just a selection of books available from Australia's largest range of computer literature.

The S-100 Bus Handbook (Bursky)	$16.50
Computer Games for Businesses, Schools & Homes (Nahigian)	$14.75
Basic Computer Games (Ahl)	$11.75
Microcomputers for Business Applications (Barden)	$11.95
Build Your Own Working Robot (Heiserman)	$7.50
How to Build Your Own Working Robot PET (Da Costa)	$8.50
Getting Acquainted with Microcomputers (Frenzel)	$11.95
Microprocessor Cookbook (Horseski)	$7.50
Z80 Software Gourmet Guide & Cookbook (Scelbi)	$20.60
6502 Software Gourmet Guide & Cookbook (Scelbi)	$16.80
Illustrating Basic (Alcock)	$6.95
PET-CBM Personal Computer Guide (Donahue)	$20.50
Stimulating Simulations (Engel)	$6.00
Introduction to the Computer (Fuori)	$12.95
6502 Assembly Language Programming (Leventhal)	$16.20
Data Processing With Applications (Condon)	$17.95
Microcomputers/Microprocessors: Hardware, Software & Applications (Hilburn)	$30.50
Minicomputer Systems (Lines)	$26.95
Microprocessors: Technology Architecture & Applications (McGlynn)	$19.00
Microprocessors & Microcomputers (Tocci)	$22.95
Handbook of Microprocessors, Microcomputers & Minicomputers (Lenk)	$24.25
Microcomputers: A Technology Forecast & Assessment to the year 2000 (Wise)	$22.35
The Art of Digital Design (Winkel)	$33.75
Computers in Business (Sanders)	$9.75
Minicomputer Systems: Organization, Programming & Applications (Eckhouse)	$30.95
Microprocessor Programming & Software Development (Duncan)	$34.95
The Basic Cookbook (Tracton)	$6.50
Pascal: An Introduction to Methodical Programming (Findlay)	$15.00
Problem Solving Using Pascal (Bowles)	$14.25
Structured Walkthroughs (Yourdon)	$33.75
Managing the Structured Techniques (Yourdon)	$33.75
Structured Design (Yourdon)	$28.50
Distributed/Micro/Minicomputer Systems (Weitzman)	$30.50
Structured Computer Organization (Tanenbaum)	$16.95
Computers in Action (Spencer)	$8.00
Crime by Computer (Parker)	$7.50
8080A/8085 Assembly Language Programming	$16.20
Architecture of Microcomputers (Greenfield)	$33.75
Syntax of Programming Languages (Backhouse)	$27.75
Structured Analysis & System Specification (De Marco)	$33.75
EDP Costs & Charges: Finance, Budgets & Cost Control in Data Processing (Cortada)	$33.75
The Mighty Micro (Evans)	$16.50
Computer Organization, Hardware/Software (Gorsline)	$26.95
Introduction to Data Processing (Popkin)	$19.95
The C Programming Language (Kernighan)	$15.95
Computer Power of the Small Business (Sippl)	$8.95
From Television to Home Computer (Robertson)	$21.80
Basic & The Personal Computer (Dwyer)	$15.95
Elementary Structured Cobol (Davis)	$18.85
A Guide to Fortran IV Programming (McCracken)	$18.55
Microprocessors & Microcomputers (Huggins)	$12.50
Minicomputers: Structure & Programming (Lewis)	$19.05
Computer Based Business Systems (Jeffery)	$7.95
Microcomputer Primer (Waite)	$10.75
Microcomputer — Analog Converter, Software & Hardware Interfacing (Titus)	$12.95
Computer in Action (Spencer)	$8.00
Microcomputer Interfacing (Artwick)	$24.25
Digital Circuits & Microcomputers (Johnson)	$25.75
The Cobol Environment (Graver)	$26.95
Successful Data Processing System Analysis (Gildersleeve)	$22.95
Programming in Basic for Business (Bosworth)	$13.35
Structured Systems Analysis (Gane & Sarson)	$26.50
Complete Handbook of Robotics (Safford)	$9.95

McGill's Authorised Newsagency Pty. Ltd.

187-193 Elizabeth Street (Opp. G.P.O.), Melbourne, 3000.
Tel. 60 1475, 60 1505

Mail Order Welcome. Credit card accepted. Postage: local $1.25, interstate $1.70

Will IBM team up with Madame Tussauds to produce 'living' replicas of our dead relatives? This chilling spectre of a future in which loneliness and depression are countered by 'plastic pals' is just one aspect of ANIMISTICS as proposed by Neil Frude — lecturer in Clincial Psychology at University College, Cardiff.

ANIMISTICS

The scene is a man alone in the evening in a large computer installation, overturning teletypes, smashing equipment, destroying irreplaceable data tapes; he's not a life-time Luddite of the new school, but a highly-trained operator with many years experience in programming and systems operation. The frustration he feels when things don't operate to plan is an exaggeration of the emotion which many people experience when faced with the repeated failure of a system or a program. Such a scene has been realised a sufficient number of times, with predictably disastrous consequences, for IBM to now be financing large-scale research by psychologists into 'user-friendliness' in micro-based systems.

American psychologists Karl Scheibe and Margaret Erwin left a tape-recorder running in a room in which subjects played games with a micro. The spontaneous comments which emerged ranged from the affectionate to the downright hostile. The machine was referred to as 'it', 'you', 'he' and 'Fred' (never as a female) and, say the experimenters, "the use of profanity was common". These psychologists concluded that the computer is very easily cast in the role of another person. Adrian Hope, writing in *Everyday Electronics* described an unconventional operation with the Texas Instruments voice synthesiser 'Speak and Spell'. To expand the vocabulary there is provision for an additional plug-in ROM, accessible by a 'module' button on the keyboard. It appears that if this is pressed when a module is not inserted then the machine invents words and phrases. "So pathetic is the garbled sound", writes Hope, "that only the hardest heart can fail to feel sorry for the confused electronics burbling as if in final, demented death throes."

Each of these emotional and 'personal' effects of machines and programs is incidental to the design of the systems involved. Such reactions are secondary, and often unwanted. If we *try* to humanise a machine then the effects are far more devastating and may be very easy to achieve. Take the simplest of 'programs' in which there is displayed on a VDU the question "WHAT IS YOUR NAME?" with provision for a string variable input. The user types in "JOHN" and the machine, using this string, then prints "THANK YOU, JOHN, NOW LET'S HAVE SOME FUN!" Now no machine is likely to run out of memory on that program, and it doesn't take a two-month programming course to write the software, yet the psychological effect on the naive user is often profound. With appropriate skill (and they are the skills of the playwright rather than of the high-grade programmer) the user, child or adult, is easily seduced into further interaction. Statements and reactions by the machine can arouse feelings of humour, affection, hostility, boredom, excitement and, in principle, the whole range of human emotions.

So far there has been relatively little interest in 'humanising' machines. Perhaps those interested in recent developments are more intrigued with the technological potential rather than the human potential of new technology. This will certainly change as machines increase in number and reach further than the 'hard core' of technologists and business systems people, as micro applications swamp into more and more fields and as the economic rewards of mass sales to the technically unsophisticated become apparent. Chips may now be invading homes in the form of calculators, television games and watches but there is a far greater potential market for pets (with the lower case 'p'). When this is realised, then we can expect the parallel development of 'micros as calculators' and 'micros as companions'. This scenario has no need to await future technological developments but would

realisation of current potential together with a leap, or several leaps, of the imagination. It needs psychologists, playwrights, technologists and programmers to cooperate to produce the viable companion. The prospect is both exciting and frightening, yet economic pressures make its realisation seem inevitable. The dream of every chip salesman should be microcircuits, warm and fur-covered, contentedly purring away in every old lady's lap, looking up once in a while, speaking words of reassurance and of its love and need for her, and reminding her to take her tablets at the right time.

If such a prospect seems laughable then we should bear in mind some of the psychological factors which will contribute to its becoming a fact. The viability of the 'intimate machine' rests on two psychological premises, the desire for (and indeed the real benefits of) intimacy and the tendency of people to treat inanimate objects possessing certain vital features 'as if' they were animals or people. These characteristics together ensure the viability, for a large number of potential customers, of the 'plastic pal', the 'micro friend'.

When social scientists conduct 'happiness surveys' to determine the correlates of happiness, and when they ask people about the most important things in their lives, it emerges that the people who are most happy are those with several friends and social contacts, particularly very close ones, and that people say that they value most highly (even above wealth and health) their relationships with other people. Psychologists have provided lists of those factors in social contact which seem to be of particular value, to help people and to make them happy. These factors, such as 'feeling close to', 'feeling responsible for' and 'feeling known by' the other person have been further analysed so that we understand something of the particular behaviours and interactions which foster such feelings. These analyses might well provide the psychological groundwork for any attempt to simulate such actions of the 'other person' in a machine form. The practical benefits of intimate contact are undoubtedly great and we can link this with the fact that those without such regular interaction seem psychologically vulnerable. Single people, the widowed and the divorced are at greater 'risk' of mental breakdown, depression, suicide and alcoholism.

Recent studies have indicated that relationships with pets may go at least some of the way towards satisfying the need for intimacy. The sad fact is that not everybody has a family 'on tap' — there are many lonely and isolated people, particularly among the old, and there is now good evidence to suggest that some people gain from their cats and dogs, budgies and tortoises many of the psychological rewards which most of us obtain from satisfactory relationships with other people. Now the limited behavioural repertoire of some of these creatures would have suggested that they would *not* be likely to prove satisfying as companions and the fact that they *do* brings us to the next psychological premise in our argument — that people tend to 'read into' creatures and objects characteristics which are typical of higher forms. This process has been labelled 'animism' and it has been the subject of considerable study by psychologists and anthropologists.

In the 1940s the psychologist Michotte built a contrivance by which two shapes were seen as coming together at various speeds and 'colliding' with various patterns of reaction. What Michotte found was that people tended to interpret these visual patterns not only in a 'causal' way (they saw a 'billiard ball effect', a 'pushcart effect' and so forth) but also a 'human' or 'animistic' way. Thus one object might be said to push another one 'deliberately' or 'viciously'. There was then a tendency to attribute motives, emotional expression and intentionality to simple moving shapes. In other experiments short pieces of cartoon film have been produced, and once again it is easy to get people to report 'high level' interpretations — they ascribe 'animistic' qualities to simply moving geometric patterns. It seems, too, that particular shapes often bring out a specific emotional response. A skilled cartoonist need only draw a few lines to create a baby rabbit or 'Bambi' figure, which is not only easily recognisable but also 'appealing' and of course doll manufacturers successfully recognise this tendency and turn it into product and cash.

It's not just visual presentations that produce such emotional reactions, either. Quality of voice, the nature of statements being made, physical warmth and softness or furriness may all produce positive emotional responses. If we combine several such characteristics then the overall psychological result is greatly magnified. Anthropological interest in animism has stemmed partly from the view that it is a characteristic feature of the 'primitive mind'. Certainly it is found most strongly in primitives and children, but that is not the end of it. The experiments described above indicate that the tendency exists, albeit in a somewhat quiescent form, in all of us. A few years ago one successful marketing company in the States launched the 'Pet Rock', an executive toy, expensively packaged and with instructions for care and feeding. The joke sold well. Scratch an executive, it seems, and you'll find a primitive. We can, however, overcome the sophistication which may normally hide the animistic tendency by matching it with sophisticated technology. Some of the possibilities here are indicated in fictional creations which capture the popular imagination. There has long been a fascination with 'humanoid' automata; they are mentioned in Homer's *Iliad* and they are the stock-in-trade of much of today's science fiction, both in print and on the screen. Stars such as R2D2, Hal of '2001' and several of the characters of the 'Hitch-hiker's Guide to the Galaxy' endear themselves to viewers and listeners by virtue of their 'personality'. Their fascination does not lie in their formidable computing power but rather in their typically 'human' utterances and foibles.

There are lessons to be learned from close examination of the characteristics of these popular creations. Almost all of these androids are conceived of as male, they are all primarily task-oriented or problem-solving machines with merely incidental personality rather than being specifically contrived humanoid companions, and they are mechanically rather primitive with a surfeit of whirring cams and flashing lights. Some of their voices are far more stilted and sound far more artificial than the best of the voice synthesisers available currently. In a word they are in many ways too 'hard' and would be unlikely, were they realised, to be immediately acceptable as companions. The problems of 'softening' the technology, however, are not difficult and are largely surmountable with currently available methods. What is needed is imagination, research into mass-user acceptability and a belief that there are likely to be vast social and economic pay-offs.

Softening the hardware

It's undeniable that there is a 'machine

barrier'. People feel initially self-conscious and uncomfortable 'relating' to a machine. The same kind of self-consciousness is often found when one is discovered talking to a cat. Yet (in private at least) some people talk to their cats all the time. If they overcome the 'animal barrier' involved in treating animals 'humanly' then they can probably also be seduced into treating machines in the same way. The 'human-ness' of sophisticatedly programmed machines with appropriate software is likely to be far greater than that of any animal, although the barrier, it is true, is likely to be more formidable, at least initially.

What hardware features would the ideal micro companion possess? What should it look like, feel like and sound like? Presumably people would relate more easily to a body shape which they were familiar with and so a human or animal form would seem to be most appropriate. A soft skin or fur covering would feel pleasant to the touch and a suitable body temperature could be maintained. Above all, the ideal companion would *not* look like today's computer, no shiny metal parts, no VDU or flashing lights. Facial features could be customised so that no two machines actually looked alike (computer controlled production would make this easy and cheap) and voices could be tailored in a similar fashion so that no two sounded exactly alike. The state of the art in voice synthesis is now adequate for this aspect of the production of a good companion though the voices produced are a bit harsh and school-masterly. The user would want less perfection, more pauses, splutters, repetitions, coughs and giggles. We would expect 50% of production to be of female voices (and we would naturally want to combine these with female body shells unless we have a consumer with rather particular needs.)

The possibility would exist for producing a model which not only looks human but which looks like a specific person, someone famous perhaps, or an absent member of the family. The psychological effects here are quite unexplored. Would schools keep an appropriately programmed Shakespeare replica in the cupboard to teach English? Perhaps there would be legislation to prevent the simulation of a person until 50 years after their death as in the existing copyright laws. The chilling thought of a lonely person sitting in conversation beside the fireside with a replica of a deceased spouse does little to assuage fears as to the possible social impact of the application of technology in the way we are envisaging. It may even give us cause to ponder the desirability of consciousness-raising in the present form; yet the elements for these developments are lying about us in separate packages and it cannot be long before somebody will put them together. There is, after all, a *lot* of money to be made.

At this stage it seems that realistic locomotion is one aspect of the hardware side of things which is not readily achievable. Maybe the first generation of companions will be relatively sedentary. Other body movements may be complex but are not difficult in principle, as witnessed by the more successful of the achievements of the automata makers in the 18th and 19th centuries.

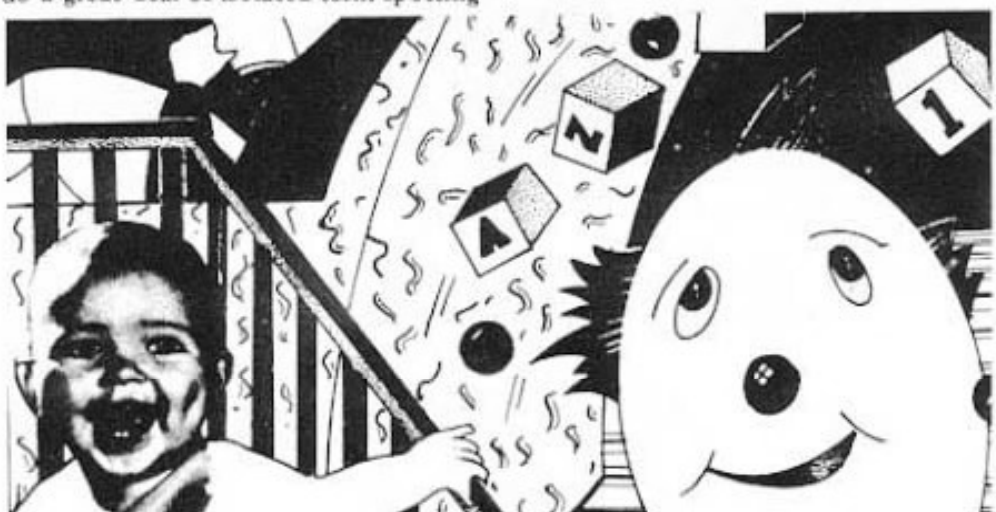
ancient history and testifies to a long-established desire to create realistic humanoids. The then 'new technology' of clock-making gave rise to a great leap forward in the production of such machines in the 17th and 18th centuries and we can expect a similar and much greater impact-making leap with today's new technology. The problem with sophisticated machines of the old era was that they were hand-produced and made on a 'one-off' basis. The mass-produced automata were far simpler toys with very limited movements. Today, of course, it's possible to produce in quantity even the most sophisticated machines with a very much extended repertoire of movement. In St. Petersburg in 1799 the Academy of Sciences offered a prize for the first machine which could realistically produce the five vowel sounds. We can imagine the contraptions which were produced, all bellows, bladders and reeds but nevertheless designed and constructed with a great deal of care and ingenuity. If yesterday's automata makers had had the opportunity to employ today's technology then their productions would have been truly astonishing. The old automata engineers were not content, however, to merely produce effective functional mechanisms; they took great pains to incorporate them into life-like models. This made them far more awesome and intriguing to a public which queued and paid to see those 'miracles of the modern age'.

Softening the software

The production of attractive and realistic dolls provides a vehicle for the output and display of the control systems which are the forte of contemporary technology. The animistic potential of an appealing voice and moving body is fulfilled only when what the machine *does* and *says* is realistic and appealing, too. Already there are successful attempts to simulate 'human' conversation, though in a limited form and via a VDU and teletype, in counselling and psychiatric programs such as Weizenbaum's ELIZA and in medical diagnostic programs. It's true that these have a very limited repertoire and generally work by searching for and recognising key terms. However, we probably overestimate the degree of complexity and the extent of the repertoire of normal social conversation. Certainly people do a great deal of isolated term-spotting and 'filling in', and we just don't know how sophisticated an informal conversational program would have to be in order to be pleasurable and user-acceptable. Conversation with young children or with the senile can be difficult and arduous but may, nevertheless, be pleasurable. The *type* of errors which the machine would produce would certainly be somewhat different from those which children make, however, and it remains to be seen whether people's reactions to these would be of the same kind. What is certain is that errors in social chatting are not of the same practical importance as in task-oriented interactions; they may be amusing and easily tolerated, or perplexing and difficult to live with. It's likely that people would accommodate to the limitations of the machine, as they do with young children and other people with low comprehension, and alter their speech patterns so that they produce statements which will be understood. There is a natural process by which linguistic style is 'shaped up' in accordance with the perceived effects of former interactions, and of course we would expect the machine software to contain the potential for a similar accommodation and 'learning from' the input style of the speaker or teletype user.

The style of informal speech is not, of course, that which we see in the typescripts of a carefully written play (unless it's by a playwright of the Pinter school) but contains much repetition, pausing, restatement and imprecision. Thousands of recordings of 'ordinary' conversations have been analysed by linguists and psycholinguists and it's not difficult to produce a simulation of the style. But such a level of analysis need not in fact be necessary and the probem might be successfully solved in a more direct fashion by the programmer with a good ear and some of the skills of the dramatist and by a program with the right degree of flexibility and randomness. Without formal analysis a 'try it and see' approach would be employed.

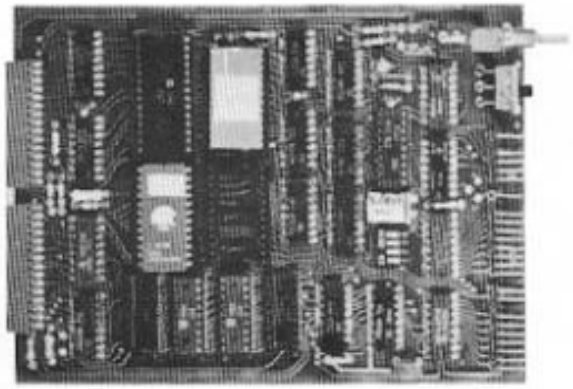
Next we come to the 'personality' of the machine as implemented in the programming. The computer simulation of personality has a relatively long history. Loehlin's program ALDOUS recognises situations, reacts 'emotionally' and in various versions is a decisive or hesitant reactor. There is also RADICAL ALDOUS, CONSERVATIVE ALDOUS and SAINT ALDOUS. We see here a good opportunity for the customisation of programs. The machine should be basically sympathetic and 'good' but



Get Serious

Two microprocessors from MicroPro Design to provide you with a choice. The tried and proven "Front Panel Computer" or our new economical "Single Board" processor. Either way you obtain the power of the 6502 microprocessor in a form that is ready to run using the on-board digital input/output lines or any combination of the range of standard interfaces available.

MicroCon
"Front Panel Computer"
$200



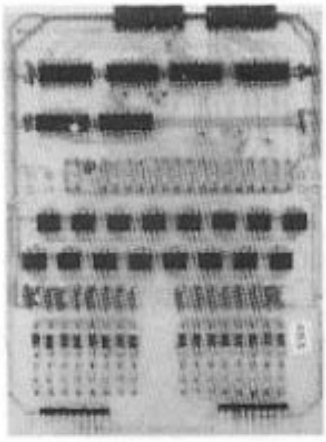
MicroCon OEM
"Single Board Computer"
$150

The MicroCon/OEM microprocessor is completely compatible with our original MicroCon. Use MicroCon as an economical development tool and when you are ready plug the programme into the OEM version to keep finished cost down.

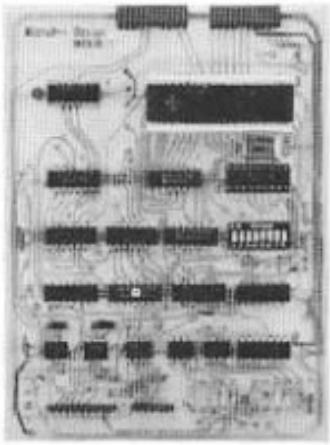
OR

With a standard RS232 terminal use the MC/OEM itself for development with our unique real-time monitor. This allows you to watch what is happening in your working programme.

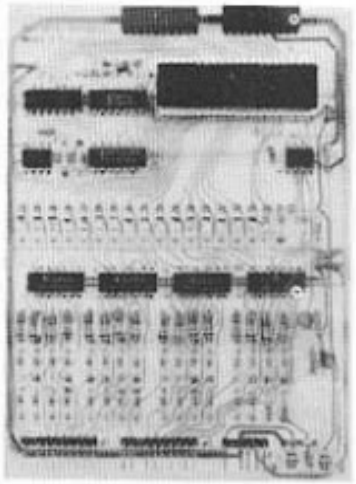
A few of the interfaces which are available to plug directly into the MicroCon system. Most cost around $100. You can even construct your own on the prototyping board which costs just $17.



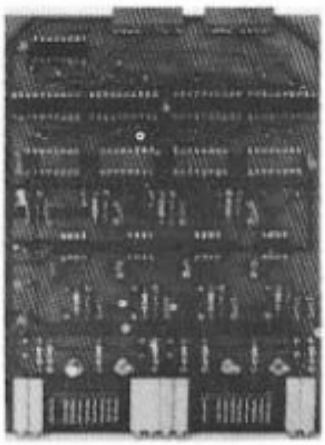
MCDI 1
Digital Input



MCSIO
Serial I/O



MCAD1
Analogue Input



MCDA2
Analogue Output

So when you are ready to get serious . . . so are we. Delivery is usually Ex-stock to four weeks. Contact us for a copy of our brochure and information about specific applications.

MicroPro Design Pty. Ltd.

P.O. Box 153, North Sydney, N.S.W. 2060 PH: 02-4381220

perhaps also have occasional moods of obstinacy or naughtiness. It should be predictable but not *too* predictable, vary through the day and accommodate to some extent to the mood of the user. It should be a willing chess opponent when required and also occasionally suggest a game, and take both wins and defeats 'in character'.

The initial contact between user and machine may well be stilted but this will change as intimacy develops. First encounters are always very important and skilful programming would be required to ensure smoothness. There are, of course, already a number of programs which are designed to overcome even an initial hostility of a user. They spot hostile statements and swear words and 'give as good as they get', attempting to win over the aggressive and unsympathetic person interacting with them. These attempts to enable the machine to 'make friends and influence people' display a good deal of the psychological insight and subtlety which may well become typical of this new area of 'soft' software.

As interaction with the user progresses, the machine should adapt and settle and become more 'at home' and more 'in tune', changing from the initial hesitancy which characterises a first meeting with a stranger to the relaxed mode which is more typical of an established friend. It should build up a representation of the user's personal world, remembering his likes and dislikes, storing some information from past interactions and recalling aspects of their previous life together. The user will thus become more predictable to the machine while at the same time the machine becomes more predictable to the user.

The practical uses of companion machines are, of course, numerous. They will act as entertainers, comforters, memory-aids, calculators, teachers, guard-dogs and telephone-answering machines. They will read aloud texts from newspapers and books, play games, make suggestions for meals and call a doctor when the user is ill. They will add interest to life, provide an opportunity for care and nurturance and be an ever-present tonic against boredom and loneliness.

Initial doubts about the viability of real 'personal contact' between human and machine may be dispelled when we see the pleasure and apparent intimacy which people show with their pets, and attribution of personality and motive on the basis of the often limited and non-interactive behaviour of these animals. The evidence suggests that, despite the 'machine barrier', within hours, or perhaps even minutes, of suitable man-machine interaction many of the initial inhibitions are readily overcome. The uniqueness of the 'experienced' machine which has been living with its user, the subtle reactiveness and personality which can be displayed, and the blend of predictability and unpredictability which will characterise these companion systems seems to offer every chance that they will more than adequately provide for many of the functions which pets, human acquaintances and maybe even intimates currently fulfil.

We may regret that anybody could have use for such a 'person substitue' yet it's undeniable that many people (and for some functions maybe *most* people) do have spaces in their lives which a suitable machine could help to fill. How the realisation of this current and near-future potential will affect the social order can hardly be imagined, but the effects will certainly be profound. We know a great deal about human needs, we know that many social needs are not being fulfilled by many people, we know many ways in which they can be satisfied and it seems probable that some at least can be satisfied by silicon. Although many people are bound to find the idea repugnant and an insult to the nature of man, the forces of the market place will ensure that the necessary links are made between existing modules and existing skills to soon produce the companion machine. The implications are devastating, we should be talking about it, we had better be prepared.

APC anticipates that as controversial an article as this one by Neil Frude is bound to provoke considerable reaction from readers. Should this be the case there may well be the opportunity to produce a 'follow-up' article, based on that response. Of particular interest will be:
(1) Contributions of short programs in the Animistics field (and maybe a prize for 'the best').
(2) Anecdotes relating to personal experiences — especially with children.
(3) Ideas on 'machine softening'
(4) Comments on the social implications of what has been suggested.

COMPUCOLOR II

MICRO COMPUTER
from $1970 tax paid

Features
- UP TO 32K USER RAM
- 5" DISK DRIVE
- 8 COLOUR VIDEO DISPLAY
- RS232C PRINTER PORT
- FULL GRAPHICS DISPLAY

MICROLINE 80

IMPACT PRINTER
from $960 tax paid

Features
- FULL ASCII UPPER & LOWER CASE
- FULL GRAPHICS
- 80 CHARACTERS PER SECOND
- 9 x 7 DOT MATRIX
- 80 or 132 CHARACTER LINE

THE LOGIC SHOP PTY. LTD.

212 HIGH STREET, PRAHRAN, 3181 (03) 51 1950

M68000 -MOTOROLA'S SWEET SIXTEEN

News has been coming in thick and fast recently about Motorola's forthcoming addition to the new range of 16-bit super micros, the M68000. But with something approaching a six month wait before any sort of general availability, has it arrived too late? Nicholas Jarmany largely dodges that question and instead casts an appreciative eye over its capabilities.

The Intel 8086 was the first of the new 16-bit micros to appear, closely followed by the Zilog Z8000. At the moment there is still no physical sign of Motorola's contender, so presumably Intel and Zilog are rubbing their hands with glee. The only dampener for them is that the M68000 is almost certainly the most powerful of the three; in fact at one stage when some of the big manufacturers saw the advance specifications, it was said that Motorola just wouldn't be able to make it. It now looks, however, as if the scepticism was ill-founded for sample devices are already spreading round the world.

Internal Operation/Layout

The M68000 internal structure is that of a 32-bit micro, making it very efficient with long word operations. There are 17 32-bit registers (apart from a 32-bit program counter and a 16-bit status register) comprising eight data registers for 8, 16 and 32 bit data and seven address registers. All 17 registers can be used as index registers and there is also a specific user and supervisor stack pointer.

There are two modes of operation, user and supervisor. In user mode certain instructions are illegal and areas of memory can be locked out by a memory management unit. When in this mode a switch to supervisor mode always occurs when an interrupt, bus error etc is received. In supervisor mode all instructions are available and the full status register can be accessed. This arrangement is similar to that of the Z8000.

A trace mode can be set in supervisor mode which causes a branch via a trace vector after execution of every instruction — very useful for program debugging! The lower 512 words of memory are reserved for a vector table containing 255 vectors of which 192 are reserved for user interrupt vectors.

Interrupts, bus errors etc. all cause what Motorola calls 'exception processing', of which there are three levels of priority. In order of decreasing priority, Group 0 contains — Reset (highest), Bus Error, Address Error; Group 1 — Trace, Interrupt, Illegal Instruction and Privilege Violation; Group 2 (all equal priority) — TRAP, TRAPV, CHK, Zero Divide. All the exceptions cause branching via the appropriate vector, except for certain occurrences of Bus Error. If a Bus Error and a Halt signal are received simultaneously, the processor will re-run the current memory access on the negation of $\overline{\text{HALT}}$.

Instruction set

There are 56 basic instruction types and 14 addressing modes, and although this doesn't seem like many instructions, it's deceptive as there are many variations. For example MOVE caters for loading data in memory etc. The total number of useful instructions exceeds 1000! The addressing modes are extremely comprehensive and no programmer could envisage needing more. The format of the instructions is astonishingly simple and easy to use. With other micros you have to learn the code for each individual instruction — e.g. Load register (indexed) might be 0A and load register (immediate) FE. Not so with the M68000 . . . All you need to learn are the numbers for the 56 basic instructions and the numbers for the addressing modes. The complete instruction is then made up of the code for the instruction, the data size, addressing mode and register number (if required). Dead simple!

Speed

The speed of the M68000 is also something to be marvelled at. It's faster than the 8086, the Z8000 and the PDP11/45 — and it can't be a lot slower than the PDP 11/70! It's twice as fast as the Z8000 on a 16-bit multiply (35 instruction cycles compared with 70 — maximum).

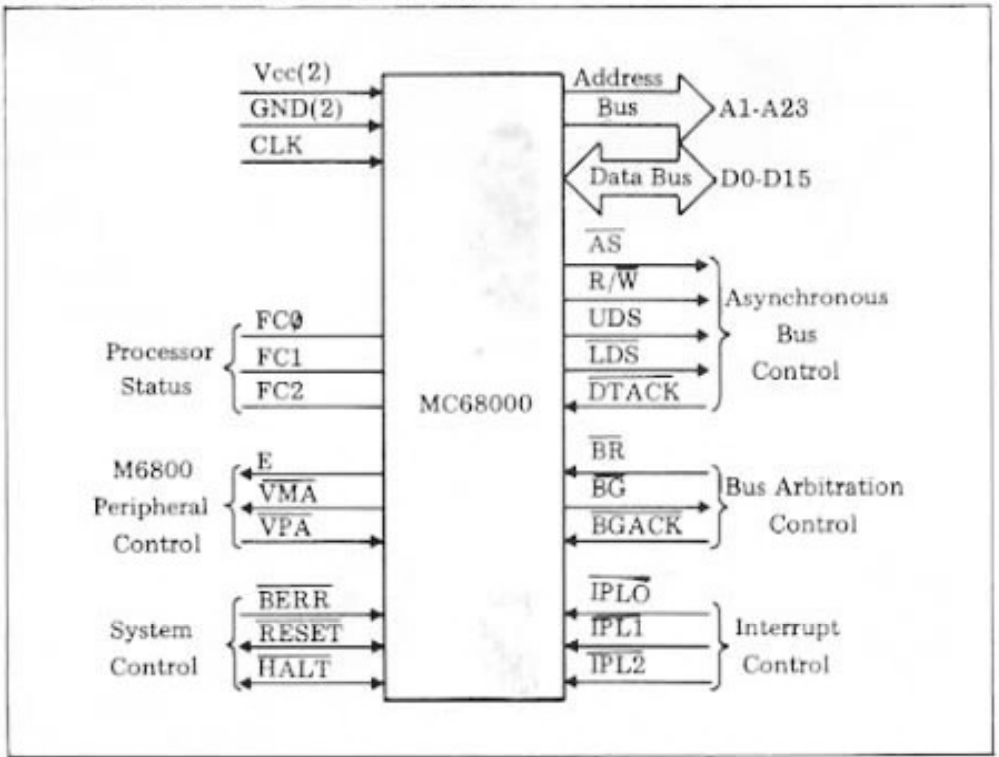
Omissions

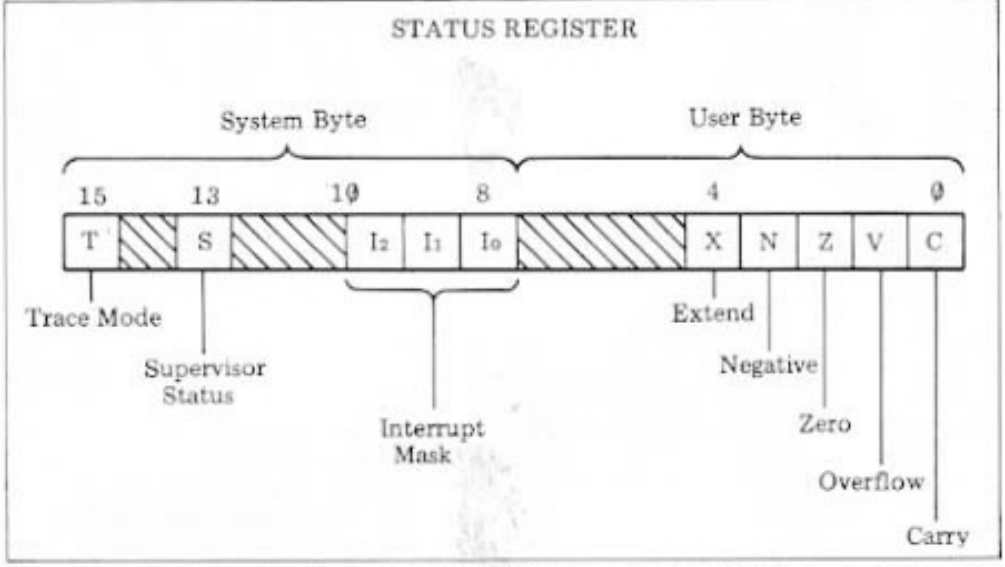
Unlike the Z8000 the M68000 does not have on-chip refresh and multi-micro control. It could be that Motorola does not want to be seen to be abandoning its traditional approach in favour of

FUNCTION CODE OUTPUTS

FC2	FC1	FC0	Cycle Type
L	L	L	(Undefined Reserved)
L	L	H	User Data
L	H	L	User Program
L	H	H	(Undefined Reserved)
H	L	L	(Undefined Reserved)
H	L	H	Supervisor Data
H	H	L	Supervisor Program
H	H	H	Interrupt Acknowledge

L = Low H = High





somebody else's. I feel, however, that there has been a preference towards getting as much computing power into the CPU as possible — at the expense of other features that can easily be added on with a few external chips. (Try extending an instruction set with a few external chips!) What Motorola has aimed at is producing the most powerful single chip CPU in the world. Possibly the plan has succeeded.

Hardware

The M68000 has definitely been designed for large systems, although a small system could easily be based around it. The processor is contained in a 64-pin package (long!) which is needed because none of the signals are multiplexed, thus increasing speed and ease of use. It requires +5V and a single phase clock (up to 8MHz). . . an internal cycle is defined as two clock cycles (250ns). There are 23 address lines giving 16 Mbytes of direct addressing; individual bytes are accessed via the $\overline{UDS}$ and $\overline{LDS}$ signals. A valid address is indicated by $\overline{AS}$ and the addressed device responds with $\overline{DTACK}$ (Data Transfer ACKnowledge). This also acts to stretch memory cycles (if necessary) by not being negated until the memory is ready. Memory read, write, read-modify-write cycles take 4, 5 and 9 clock cycles respectively. A great feature of the M68000 is its ability to interface directly with standard M6800 peripherals. If, at the beginning of a memory access cycle, a $\overline{VPA}$ signal is received, the processor switches to the M6800 form of addressing. $\overline{VMA}$ is taken low and E is equivalent to M6800 $\phi 2$ (1MHz). Hence $\overline{VPA}$ can be derived from the address decoding logic on any M6800 peripheral boards.

FC0 to FC2 are outputs that show the type of processing currently being done in the CPU; $\overline{IPL0}$ to $\overline{IPL2}$ are inputs devoted to interrupts. Seven levels of interrupt are available (level 0 = no interrupt), level 7 being the highest priority. With all seven levels of interrupt the vector address for the service routine can either be supplied by the interrupting device — or else an autovector can be used. To my knowledge this is the most advanced form of interrupt handling available on a micro. On reception of an interrupt, an interrupt acknowledge code is placed on FC0-FC2 and a read cycle is entered with the interrupt level on the lower three bits of the address bus. The processor then expects the vector address to be placed on the data bus and $\overline{DTACK}$ to be given. If this does happen then the processor jumps to the location pointed to by the contents of

DATA ADDRESSING MODES

Mode	Generation
Register Direct Addressing	
Data Register Direct	EA = Dn
Address Register Direct	EA = An
Absolute Data Addressing	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA = (PC) + d_{16}
Relative with Index and Offset	EA = (PC) + (Xn) + d_8
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (An) An ← An + N
Predecrement Register Indirect	An ← An - N, EA = (An)
Register Indirect With Offset	EA = (An) + d_{16}
Indexed Register Indirect With Offset	EA = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA = SR, USP, SP, PC

NOTES:
EA Effective Address
An Address Register
Dn Data Register
Xn Address or Data Register used as Index Register
SR Status Register
PC Program Counter
d_8 Eight-bit Offset (displacement)
d_{16} Sixteen-bit Offset (displacement)
N 1 for Byte, 2 for Words and 4 for Long Words

INSTRUCTION SET

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
BCC	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DBCC	Test Cond, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
SCC	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test

on the bus. If this does not occur then the CPU assumes an autovector and jumps using the autovector corresponding to the interrupt level.

BERROR is an input that can signify a non-responding device or an illegal access determined by an external memory management chip. The effect this signal has depends on certain conditions already described. Both the RESET and the HALT lines are bi-directional, allowing external devices to be reset via the reset instruction. An internally generated halt is caused when a Bus Error signal is received on two consecutive memory accesses. When this occurs an externally generated reset is required to restart the CPU. This feature provides useful protection in the event of a catastrophic system failure!

Summary

The strong points of the M68000 seem to be its simple, easy to learn instruction format and its useful range of instructions (including control of both stacks and queues), coupled with the ability to maintain linked stacks. With floating point instructions on the way, writing high level languages will be a piece of cake! This sort of instruction should enable more efficient programming and the introduction of many mainframe techniques.

Bus arbitration is also very comprehensive, allowing simple control of a multi-master bus. The direct interface to M6800 peripherals must appeal to a lot of people, as it will mean that most of their existing equipment could easily be used in a M68000 system, thus eliminating a lot of annoying and expensive duplication of costs.

And yet all this extra power results in no extra difficulties in system design. More and more of the complications of circuit design seem to be disappearing into fewer LSI and VLSI chips.

Motorola has made it clear that it expects to extend the instruction set in the near future, to include instructions like FIX and FLOAT (floating point to integer and vice versa); it also expects to bring out a 16MHz version. But what about now? Small quantities of the M68000 are expected on the market in the next few months, but at the moment all the chips are going to the big firms for evaluation — not surprisingly in view of the new competition in this extremely valuable market. However, as soon as second-sources get into production the supply position is bound to improve. I was told by one of Motorola's distributors that they had achieved 98% functional chips from the very first masks — an astounding achievement for a chip of this complexity!

A memory management chip is also mentioned in the advance spec on the M68000 but no one could tell me anything about it, so I doubt if it can be appearing at all this year. But who cares? If you can get your hands on an M68000 you'll be too heavily occupied to think about anything else for some time to come!

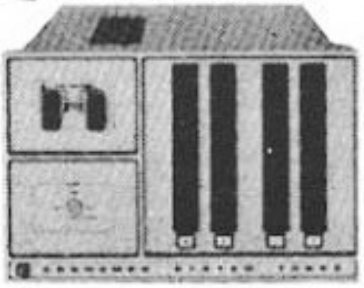
My thanks to Hawke-Cramer for helping to obtain information for this article. Technical details are based on data derived from Motorola Advanced Specification Data Sheets.

VARIATIONS OF INSTRUCTION TYPES

Instruction Type	Variation	Description
ADD	ADD	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	AND	Logical And
	ANDI	And Immediate
CMP	CMP	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	EOR	Exclusive Or
	EORI	Exclusive Or Immediate
MOVE	MOVE	Move
	MOVEA	Move address
	MOVEQ	Move Quick

Instruction Type	Variation	Description
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	NEG	Negate
	NEGX	Negate with Extend
OR	OR	Logical Or
	ORI	Or Immediate
SUB	SUB	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

Cromemco incorporated
Tomorrow's Computers Today



In the microcomputer field, the Cromemco System Three and Z-2H Winchester Hard Disc Systems stand alone in the range of features and capabilities offered. These systems are based on the Z-80A chip, and have from 1-4 mbytes of diskette storage, and from 10-80 mbytes of hard disc storage, combined with the widest range of software available in the industry, including Multi-user, Multi-tasking operation.

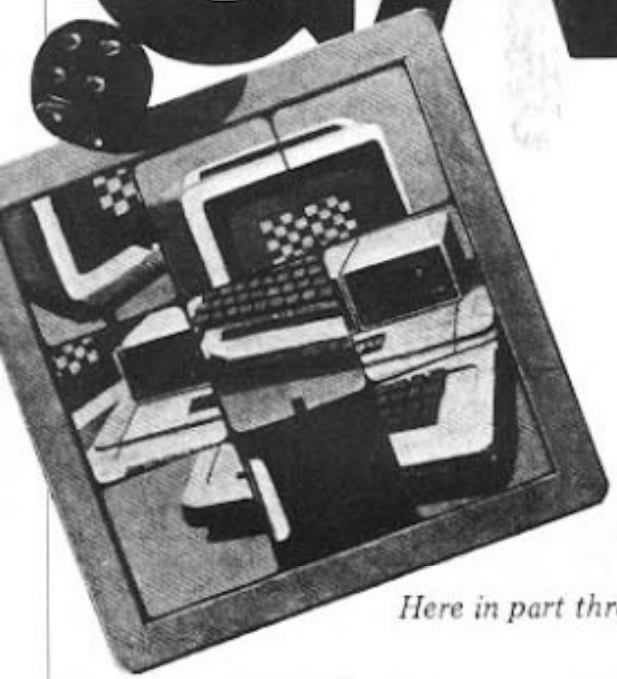
The computers have a large S100 motherboard and the operating system is a Superset of CP/M, thus allowing a wide range of non-cromemco hardware and software to be used. This also provides "obsolescence insurance". Some of these features include high resolution colour graphics, Eprom programmers, remote terminal emulation, and card reader interfaces.

Cromemco Basic, available in 3K, 16K, and 32K structured/KSAM versions, is fast, efficient, and ideal for teaching purposes because of it's dynamic error trapping on entry, and easy file handling. Cromemco Fortran IV and Cobol are equal in power to those found on mainframes, and of course, Pascal, C, and other high level languages are also available.

Informative Systems, Cromemco's authorised centre for sales and service, have installed many systems throughout Australia, backed by Cromemco trained technical staff offering maintenance, support and user training.

INFORMATIVE SYSTEMS Pty Ltd
Specialists in professional microcomputers and high performance computer support products.
3 Bank Street, South Melbourne, Vic 3205
Sydney (02) 690 2161
Telephone (03) 690 2284 Telex 30458 INSYST

COMPUTER GAMES



PRUNING BIG TREES

Here in part three of the series, David Levy introduces a minimax refinement known as the alpha-beta algorithm.

Games with big trees

Last month we discussed the use of the minimax method to search game trees, using noughts and crosses as our example. This is a game with sufficient symmetry to reduce the number of essentially different moves at the start to three: the centre, a corner and the middle of an edge. At the second ply there are a total of 12 essentially different positions, so with only seven spaces then remaining there will be an upper bound of 12 x 7! on the total number of terminal positions in the whole of the game tree. In practice the total will be somewhat less than this figure, since a number of paths will lead to a win for one side or the other, or a draw (i.e. a position in which every row, column and diagonal has at least one "0" and one "X" in it), before all nine elements of the 3 x 3 array have been filled. In order to play a perfect game of noughts and crosses with the crudest of evaluation functions, we could search the game tree exhaustively, using a score of +1 for a variation won by the program, −1 for a variation won by the opponent, and 0 for a draw.

Most interesting two-person games have much larger trees than this: in chess there are roughly one million terminal positions in an average 4-ply search, in Go the figure would be ten thousand million for a 4-ply search at the start of the game. How can we cope with such gigantic combinatorial growth in our game trees? The answer lies in a refinement of the minimax method known as the alpha-beta algorithm.

The alpha-beta algorithm

The alpha-beta algorithm owes its power to the argument that if a player can choose from a number of moves, once he finds one move which serves his purpose he need not examine the remainder of the moves in that group. Let us look at a simple two-person game tree to illustrate this point (Fig. 1).

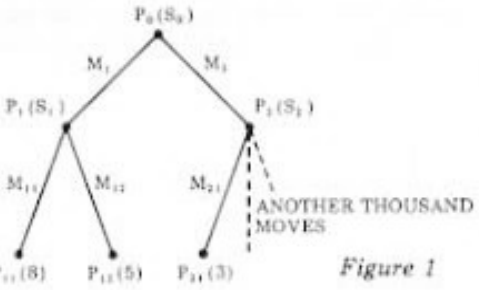


Figure 1

We shall assume that a program searches the tree from left to right, and that the evaluation function assigns scores of 8, 5 and 3 respectively to the terminal nodes P_{11}, P_{12} and P_{21}. If the program is to move from position P_0, it first considers move m_1 and then tries to decide what its opponent will do from position P_1. The opponent may choose between scores of 8 and 5, and since we have adopted the convention that the opponent's target is a low score, the opponent will choose position P_{12} with a score of 5.

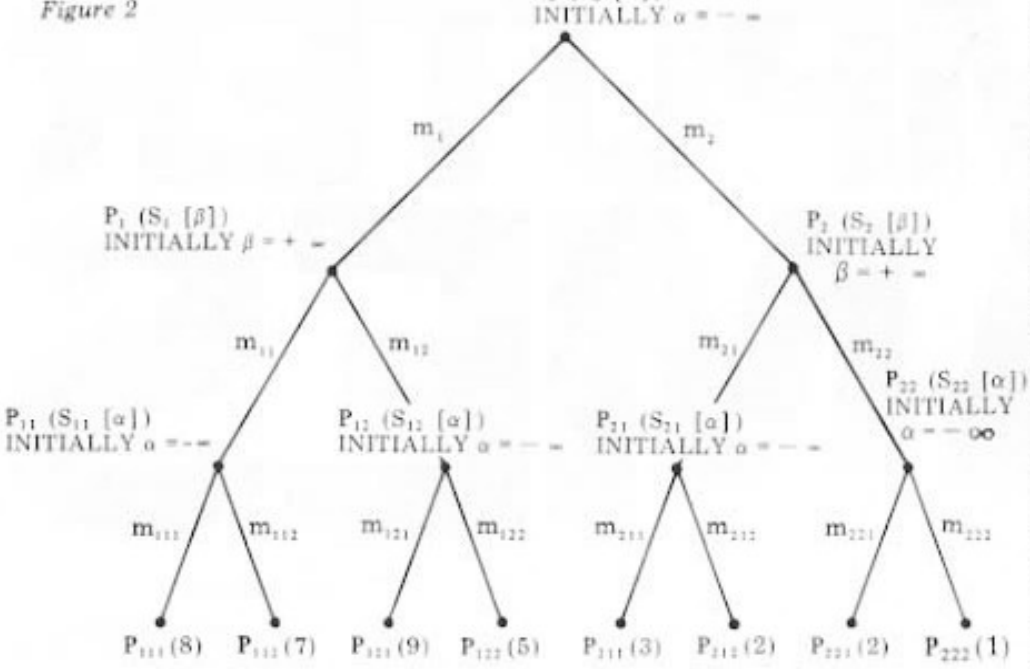
The program now knows that if it it from achieving a score of more than 5. This value of 5 is therefore the value of position P_1, assuming correct play by the opponent, and so the value 5 is assigned to S_1. We call this process of assigning values as the program backtracks up the tree "backing-up".

The score at S_1 is now backed up to S_0 and the program then considers position P_2, to determine whether it will prefer to play move m_1 or m_2. It sees that from position P_2 its opponent can, if he wishes, move to P_{21} for a score of 3, and since 3 is better than 5 from the opponent's point of view, the program will wish to deny its opponent this option and it will not, therefore, choose move m_2. It is completely irrelevant what the scores are for the thousand of unexamined brother nodes, P_{22}, P_{23}, . . . P_{21001}, because the move m_{21} is already know to refute m_2. Thus the program has determined that m_1 is better than m_2, even though it has examined only 3 of the 1,002 terminal nodes of the tree!

Of course this particular example has been specifically designed to sell you the alpha-beta algorithm, and most game trees do not allow us to get away so lightly, but the savings achieved with this algorithm are certainly substantial enough to make alpha-beta an almost essential segment in any program that searches two person-game trees. The algorithm always chooses the same move that would be selected by the minimax algorithm, but usually in a fraction of the time.

Since alpha-beta is so very important

Figure 2



for including another, more complex example (Fig. 2). This will show how the method words for a 3-ply tree and will illustrate why it has been given its strange name.

Initially, all non-terminal nodes at even ply are assigned the value $-\infty$ (α). All non-terminal nodes at odd ply are assigned the value $+\infty$ (β). As usual it is the program's turn to move from the root position P_0, and the program is trying to maximize the value of α. The opponent moves from positions P_1 and P_2, trying to minimize the value of β. The program moves from the positions at ply−2 (P_{11}, P_{12}, P_{21} and P_{22}), trying to maximize α.

The tree search now proceeds as follows:

1 Examine P_{111}. The score of 8 is greater than $-\infty$ so α at S_{11} is set to 8. This score is then compared with β at S_1 and found to be less than $+\infty$, so this value of β is also set to 8. In order to decide whether the program might be willing to play m_1, this score of 8 at S_1 is compared with $-\infty$ at S_0 and found to be greater, so α at S_0 is set to 8.

2 Examine P_{112}. The score of 7 is less than α at S_{11}, which is now 8, and since it is intended to maximize α, the value of α at S_{11} is not adjusted, and therefore the value of β at S_1 and that of α at S_0 also remain unchanged.

3 Examine P_{121}. The score of 9 is greater than $-\infty$, so α at S_{12} is set to 9. This score is then compared with β at S_1 and found to be greater, and since it is intended to minimize β the program can reject move m_{12}, knowing that its opponent can do better with move m_{11}.

4 The left hand side of the tree has now been examined and the search proceeds to the comparison of the best score achieved so far (8) with whatever can be reached, assuming best play by both sides, if the program should choose m_2. This part of the search commences with an examination of P_{211}, which is found to have a score of 3. This is compared with α at S_{21} and found to be greater, and since it is intended to maximize α the program will set this value of α to 3.

5 Examine P_{212}. The score of 2 is less than 3, so α at S_{21} (currently 3) is left unchanged, since it is intended to maximize α. This score of 3 is then compared with β at S_2, found to be lower, and since it is intended to minimize β this value of β at S_2 is set to 3. Finally this value of 3 is compared with α at S_0 (currently 8) and found to be lower. Since it is intended to maximize α, the program already knows that m_2 is inferior to m_1, because playing m_2 is not consistent with maximizing α.

The search is now over and it can be seen that only five of the eight terminal nodes needed to be examined. If you wish to verify the validity of this process by practical means, try assigning different sets of values to positions P_{122}, P_{221} and P_{222}, and you will always find that the program prefers move m_1 to move m_2.

How powerful is the alpha-beta algorithm?

During the past few years there has been considerable research into the question of just how big are the savings achieved using this algorithm rather than simple minimax. A full discussion of the theoretical and practical results of this research is well beyond the scope of this series, but the studious reader will find this work well documented in the bibliographic references found at the conclusion of this article. What follows is a summary of the most important results, and a brief discussion of their significance.

Monroe Newborn has investigated the power of the alpha-beta algorithm when searching game trees in which the

Table 1

	2-ply search		3-ply search	
b	total terminal nodes	expectation	total terminal nodes	expectation
2	4	3.67	8	6.84
4	16	12.14	64	40.11
8	64	38.65	512	220.37
16	256	122.11	4096	1214.45

in a random order. Table 1 shows, for various branching factors (b), the number of terminal nodes which we would expect a program to examine, using alpha-beta, in searches of 2 and 3-ply.

It will be seen that as the branching factor increases, so the proportion of nodes that can be ignored thanks to the alpha-beta algorithm also increases. And as the depth of search increases the effect of the algorithm is again increased. So the bigger the tree becomes, the greater will be the savings using the alpha-beta method.

The savings become even more dramatic when the branches of the tree are examined in an intelligent order. In general it is true to say that within any group of moves the best one should be examined first, so that if the best one is not good enough we need not waste time in examining the second best, third best and inferior moves. If the tree is searched in such a way that the moves are examined in their optimal order, then the number of terminal nodes examined will be approximately $2 \times \sqrt{N}$, where N is the total number of terminal nodes on the tree. Thus, for a game of chess in which the branching factor is typically 36, the number of terminal nodes on the tree is 36^4 for a 4-ply tree. Yet by using the alpha-beta algorithm, if the tree is optimally ordered we need examine only 2×36^2 terminal nodes before we find the best move from the root of the tree, a saving of well over 99% when compared with the simple minimax method.

Taking the figures from Newborn's results quoted above, we can compare the expected number of nodes examined with random ordering and the number of nodes examined with optimal ordering (Table 2).

I hope that the reader is now convinced that for all two-person game trees, except the smallest of the small, alpha-beta is a must. The most important implication of these results is that if it is at all possible, you should generate and/or examine the moves within any group or family in such a way as to take maximum advantage of the savings that can be achieved, and this means ordering the search in some way. We shall discuss various techniques for speeding up the alpha-beta search in our next month's article, but one obvious method can be mentioned here. First, generate all the moves at the root of the tree, m_1 m_2 . . . etc., and evaluate the resulting positions with the evaluation function. Sort the moves so that the move with the highest score will be examined first, then the move with the next highest, and so on.

Next look at the first position on the list and generate its successor positions. These are assigned scores using the evaluation function and they are then sorted, this time with the lowest scored

position coming at the top of the list and the highest scored position at the bottom. (This is because the program's opponent is trying to minimize the score.)

This process is repeated all the way down the tree, except for the terminal nodes, which are not sorted. Now, when searching the tree with the alpha-beta algorithm, the tree will be found to be much nearer an optimally sorted tree than if this process had not been applied. One disadvantage of this method, however, is that it requires us to keep in memory all the successor nodes to each node on the principal variation, apart from the terminal nodes. So in a search of a chess tree, with 36 moves at each node, this method would require us to keep in memory:

a the root node
b 36 nodes at each level of look-ahead apart from the terminal node.

In order to combat this problem we might try to find an extremely compact method of representing a position, but if this compactness results in a slowing down of the search process while each position is unravelled or created, much of the effect of the fast alpha-beta algorithm will be lost. Such problems require careful thought and it is often necessary to experiment before the best balance is achieved between representation and optimality of search.

Other useful techniques for examining the moves in a sensible order can often be found by thinking a little about the nature of the game. Let us consider once again the game of noughts and crosses. The elements of the 3 x 3 array might be numbered as in the following diagram:

```
1  2  3
4  5  6
7  8  9
```

A simple way to generate all the legal moves from any position is to look at the elements, starting with 1 and working up to 9, and putting any empty space on the move list. But with a basic knowledge of the strategy of the game we can speed up the search process by looking first at element 5, then 1, 3, 7 and 9, and finally at 2, 4, 6 and 8. This method of move generation takes no longer than 1, 2, 3, 4, . . . 9, yet it enables the alpha-beta algorithm to examine the moves in a more sensible order, thereby taking us closer to an optimal search process.

Next month we shall examine a flow-chart for the alpha-beta algorithm and look at further ideas for speeding up the search process.

Table 2	2-ply search		3-ply search	
b	random	optimal*	random	optimal*
2	3.67	3	6.84	5.66
4	12.14	7	40.11	15
8	38.65	15	220.37	44.248
16	122.11	31	1214.45	127

*The approximation $2 \times \sqrt{N}$ referred to above is made slightly more accurate by subtracting 1. This is not important for very large trees but it has been done here for the sake of accuracy.

Task for the month

Write a program to play noughts and crosses, taking advantage of symmetry and employing the alpha-beta algorithm. Search the whole game tree using the primitive evaluation function described above (+1 is a win for the program, −1 a win for the opponent and 0 a draw).

Test the program (a) when the moves are generated in a random order; and (b) when the moves are generated in the order: centre, corners, middle of edges. The results should indicate a useful improvement with ordered search over random search.

Bibliography

Knuth, D.E., and Moore, R.W.: *An Analysis of Alpha-Beta Pruning.* Artificial Intelligence, vol. 6, pp. 293-326, 1975.

Newborn, M.M.: *The Efficiency of the Alpha-Beta Search on Trees with Branch-dependent Terminal Node Scores.* Artificial Intelligence, vol. 8, pp. 137-153, 1977.

LOOKY VIDEO

INDEPENDANT IMPORTER/ MANUFACTURER

OHIO SCIENTIFIC COMPUTERS

HARDWARE:

1. MEMORY EXPANSION BOARD, 8K. The Double sided Plated through the Board has 8K of wired sockets with driver and buffer I.C.'s and R.F. filters. Add 2114's as you require RAM. .KIT $120.00
Fully assembled and tested, extra $ 15.00
2. 4K RAM Memory Expansion Chips, 2114, 450nS. . . . . .$ 52.00
3. R.F. Modulator, 5 to 9 Volt, all channels . . . . . .$ 19.95
4. SUPERBOARD II Cover/VDU Stand . . . . . .$ 18.95
5. SUPERBOARD II COMPUTER 8K ROM, 4K RAM $389.00
6. CHALLENGE C1P COMPUTER 8K ROM, 4K RAM $489.00
7. CHALLENGER C4P COMPUTER, Sound/Colour NTSC Colour. $946.20
Available soon — P.A.L. conversions for . . . . . .$ 80.00

SOFTWARE:
We regret, PRICE RISE, due to shipping costs. All previous (54) advertised Cassette Software, up to $2.00. Instructions (14 sets), are still the same price. O1 is now replaced by an updated, and enlarged C1 at $2.95

NEW PROGRAMMES for OHIO — SOFTWARE:

UTILITIES:
U.15. Dissassembler — 4K $10.95
U.16. Filename, for cassettes . . $ 8.95
U.17. Trace $12.95
U.18. Packer — (K saver). . . . $14.95
U.19. Cursor — C2/4 . . . . . $13.95

INSTRUCTIONS:
I.11. Dissassembled ROM, with Comments $12.95
I.12. 32x64 Character Display. . $12.95
I.13. Wp 6502 Word Processor Book . . . . . $ 3.95
I.14. Dissassembled ROM, with Inbuilt References $13.95
I.15. Sound — S II/CIP . . . . . $ 6.95
I.16. G.T. Conversion, X2 . . . . $ 2.95
I.17. Saving Data on Tape . . . . $ 6.95
I.18. CIP Tape Control . . . . . $ 6.95
I.19. CIP Beeper. . . . . $ 5.95
I.20. CIP/Base 2 Printer . . . . . $ 6.95
I.21. Morse Code Converter and Tape $16.95
I.22. 32x64 Character Supplement to I12. 100% hardware Method. . . $ 4.95

BUSINESS:
B.4. Savings and Loan Package . $14.95
D.2. Flashboard, your advert . . $10.95

GAMES:
G.25. Alien Invaders (Like Space Invaders) . . . . . $ 8.95
G.26. Orbital Lander $11.95
G.27. Escape from Mars, 2 tapes $18.95
G.28. Death Ship, 2 tapes $18.95
G.29. Startrek . . . . . $ 7.95
G.30. Air Sea Battle . . . . . $ 8.95

BOOKS/MAGAZINES/CATALOGUE
T.1. The First Book of OSI . . . $19.95
T.2. Aardvark Journal (6 issue) . $12.95
T.3. Basic Handbook, David Lien $15.95
T.4. Personal Computing Monthly $ 1.95
C.1. Catalogue describes all software and hints and free progs . . $ 2.95

Postage:
1 or 2 $1.00 — 3 – 5 $1.50
6 – 9 $2.00 — 10 or more $2.50

All prices are subject to change without notice.

All prices INCLUDE Sales Tax.
More new lines of software available — enquire.

MAIL ORDER: Check or Bankcard
Looky Video, P.O. Box 347, Richmond, 3121. Victoria.
Shop: 418 Bridge Road, Richmond, Phone (03) 429 5674

bankcard welcome here

PLOTTING IN THREE DIMENSIONS

Malcolm Banthorpe offers a simple program for evolving three dimensional representations of trigonometrical and other functions on any computer which can plot graphics to a reasonable degree of resolution. The program listing shown is written for an ITT 2020, but can be used on many other computers with very little modification.

The program originally evolved out of an investigation into possible means of representing three dimensional curved surfaces on a VDU. A wide variety of functions can be plotted as long as their range of values is restricted, as descibed later. The results are frequently surprising (at least to a non-mathemetician like me), sometimes beautiful (having an almost organic form) and nearly always interesting.

To understand how the program works, imagine a disc (Fig. 1) crossed by a series of parallel chords. If the disc is viewed obliquely it can be represented in two dimensions by an ellipse (Fig. 2). The vertical displacement f of point P from the chord AB is a function of the distance r of the point Q (which lies on AB) from the centre of the disc. Now (if you're still with me), imagine a series of such points plotted along each chord, f always being a function of r, e.g. f = sin (r). The result would be a family of overlapping curves. The program determines whether or not each point *would* be visible *if* we were actually viewing a three dimensional surface. Reference to the photographs should help to clarify how this actually works out in practice. As you can see, the program uses a series of curves parallel to the frontmost half of the circumference of the disc rather than a series of chords. This simplifies the programming and gives an arguably better display.

If you've been able to follow the explanation so far, the program listing

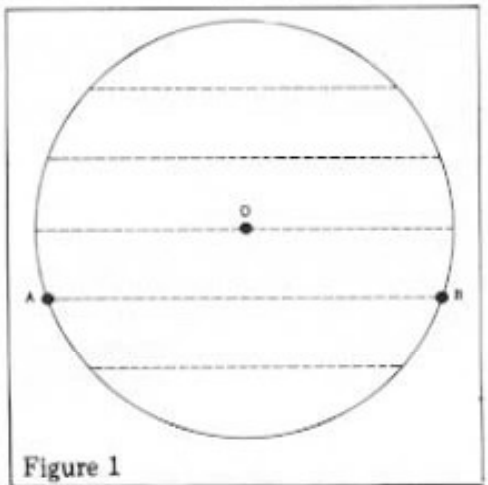


Figure 1

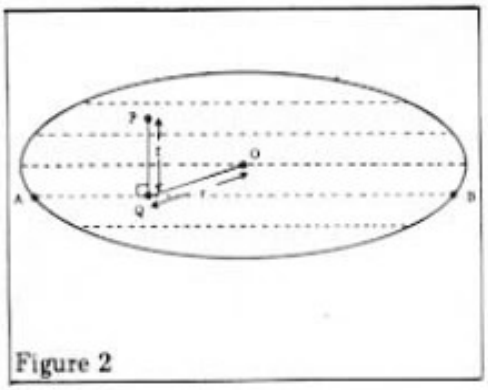


Figure 2

should be more or less self-explanatory. The function to be plotted is written as line 80 and can be changed as required. Line 75 ensures that R will always lie somewhere in the range 0 to 1. Care should be taken so that F will evaluate within the range -1 to 1 as R varies from 0 to 1.

Notes for use on other computers

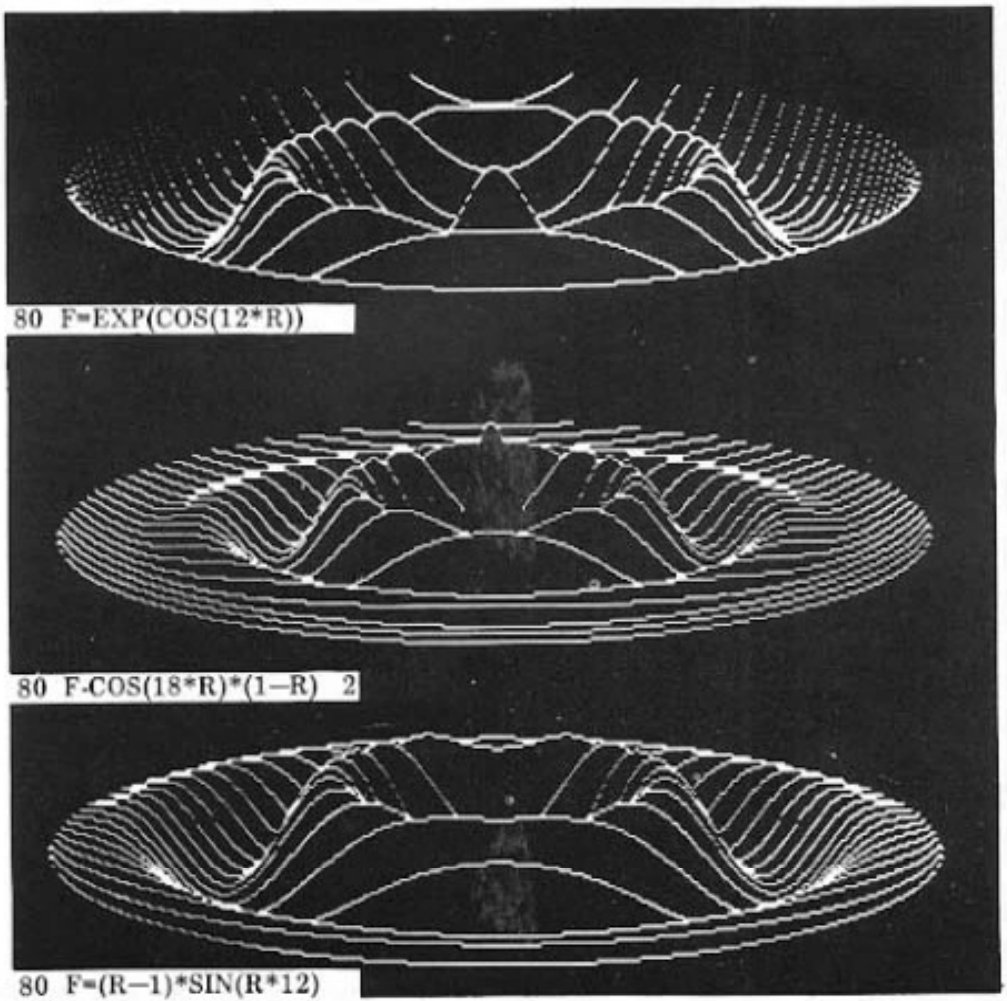
The program has been written as far as possible in 'standard' BASIC. The only possibly unfamiliar terms are as follows:

HGR sets high resolution graphics mode

HOME clears the four line text area of the screen

HCOLOR = 3 sets the plotting colour to white

VTAB (24): List 80 prints the function being plotted, at the bottom of the screen. (This may be omitted, parti-



cularly if resolution is limited, so that the whole screen can be used for plotting.)

HPLOT is equivalent to PLOT or SET in other versions of BASIC. On computers lacking such a function, it will be necessary to write a routine to POKE a character to the appropriate screen location.

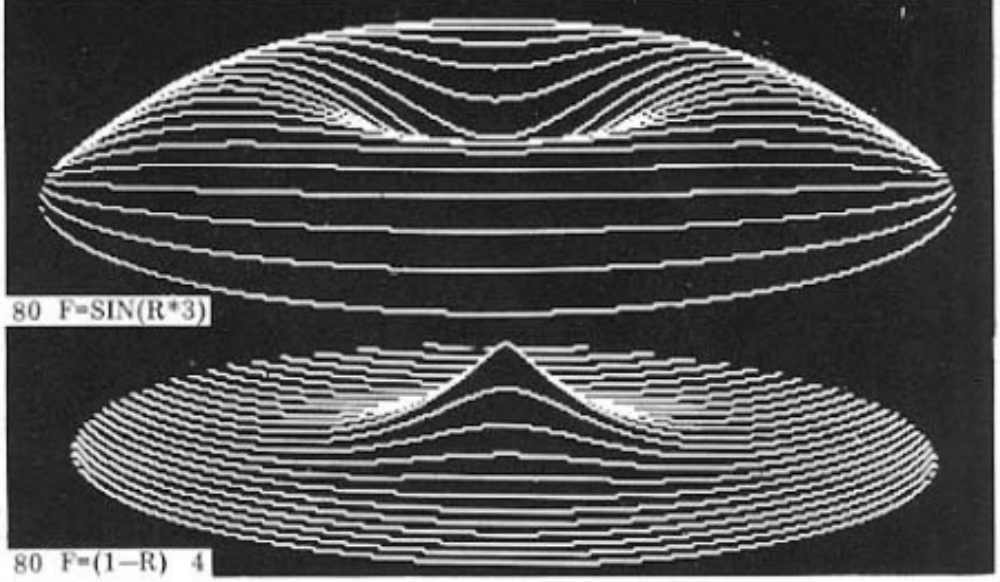
H and V in line 10 must be set to the horizontal and vertical resolutions of the system. Hence to run the program on an Apple II the only modification required is to set H to 279 in line 10.

As mentioned earlier, the program can run on many different computers but obviously the higher the resolution of the graphics, the better the display. A TRS-80 should give worthwhile results and tests with ITT's low resolution mode (40 x 40) have indicated that PET should also be suitable for experimentation, particularly if a 'double density' (80 x 50) routine can be employed. On lower resolution systems best results will be obtained plotting simple curve functions.

Line 105 assumes that the point with coordinates 0,0 is at the top left hand corner of the screen. If, as on some systems, it is at the bottom left-hand corner then line 105 should be changed to:

105 M = Y: Y = Y1 + Y

This will prevent the image from being 'upside down'. However, as the surface displayed has no objective reality outside the computer, the question of just which side should be 'up' is open to debate. Have fun . . . and I'd be interested to see the results of any further experimentation.

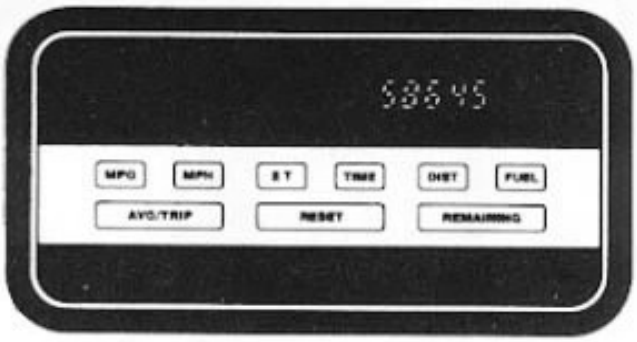



```
  1   REM THREE-DIMENSIONAL PLOTTER
  2   REM COPYRIGHT  MALCOLM BANTHORPE 1980
  3   REM
 10   HGR: HOME: HCOLOR = 3: H = 359: V = 159
 20   VTAB(24): LIST 80
 30   X1 = H / 2: X2=X1 * X1: Y1 = V / 2: Y2 = V / 4
 40   FOR X = 0 TO X1
 50   X4 = X * X: M = —Y1
 60   A = SQR ( X2 — X4)
 70   FOR I = —A TO A STEP V / 10
 75   R = SQR ( X4 + I * I ) / X1
 80   F = ( R — 1 ) * SIN ( R * 12 )
 90   Y = I / 5 + F * Y2
100   IF Y <= M THEN 120
105   M = Y: Y = Y1 — Y
110   HPLOT X1 — X, Y: HPLOT X1 + X, Y
120   NEXT I : NEXT X
130   END
```


MONITOR FUEL CONSUMPTION CONSTANTLY

with the ZEMCO 28, the latest on-board computer from the manufacturers of the CompuCruise.

A true, programmable computer, calibrated to your vehicle, giving information on fuel usage, fuel remaining, time and distance to go until empty, time on trip, distance travelled and fuel used on trip, and more. Easy, push button operation, quick, simple installation.



To: Antelope Engineering Pty. Ltd.
P. O. Box No. 271
MILSONS POINT, N.S.W. 2061

YES, please send me further information on ☐ CompuCruise ☐ Zemco 28

My name is .....

and I live at .....

..... Post Code. ..... CWM

For further information on the ZEMCO range of on-board computers, and the name of your nearest dealer, call or write to:

ANTELOPE ENGINEERING PTY. LTD.,

68 Alfred Street, Milsons Point, N.S.W. 2061

Phone: (02) 929 4033 Telex: 24432

Do it today, for more information at your figertips than you have ever had before.

THE VERSATILE ONE

COMMODORE BUSINESS SYSTEMS WITH THE ANSWER TO YOUR PROBLEM.

DEBTORS
STOCK CONTROL
INTEGRATED DEBTORS / STOCK CONTROL
COST ESTIMATION
GENERAL / CREDITOR / DEBTOR LEDGER
WORD PROCESSING
PRICE LISTS
RANDOM SORT
PLANNING
REAL ESTATE RENT ROLE

FOR SYSTEM DESIGN AT A REALISTIC PRICE CONTACT :—

THE COMMODORE COMPUTER CENTRE

bankcard welcome here

MICROCOMP
MICROCOMPUTER SYSTEMS DESIGNERS

B.S. MICROCOMP,
4th floor,
561 Bourke Street,
MELBOURNE, 3000.
Tel: 614 1433 / 614 1551

ADE

THE PROFESSIONAL'S CHOICE

COMPUCOLOR II

SPECIFICATIONS
- Integral Mini Disk Drive Standard
- Color Graphics on 128 x 128 grid
- Extended Disk Basic
- Eight colours (foreground and background)
- Up to 32 K/B user memory
- 8080 assembler available

MICROLINE 80

SPECIFICATIONS
- 80 cps 9 x 7 dot matrix
- Program selectable fonts
- Graphics
- 80-132 column printing
- 200,000,000 ch. head life
- full 96 ch ASC II Set

APPLICATIONS
Education: Programmed learning
Small accounting or business systems
Process or control functions
A multitude of games available.

DEALER ENQUIRIES WELCOME

ANDERSON DIGITAL EQUIPMENT PTY. LTD.

THE VIABLE ALTERNATIVE

P.O. Box 322, MT WAVERLEY, VIC. AUST. 3149 Phone (03) 543 2077, P.O. Box 341, Thornleigh, N.S.W. AUST. 2120 Phone (02) 848 8533. Adelaide: 79 9211. Perth: 325 5722. Hobart: 34 4522. Brisbane: 350 2611. Darwin: 81 5760. Canberra: 58 1811. Newcastle: 69 1625. Albury/Wodonga: (062) 2761. Barnawartha 129. N.Z. Wellington: 693 007. Auckland: 87 6570. Christchurch: 79 6210. New Guinea Lae: 42 3924.

10 PART PASCAL SERIES

THE COMPLETE PASCAL

BY SUE EISENBACH AND CHRIS SADLER

CHAPTER 3 CONTROL STRUCTURES: 1. LOOPS

In the last chapter, the procedure was presented as a means of performing the repetitive tasks so often required in computer programming. Thus program WALKING executed in "steps" LEFT and RIGHT alternately by successive calls to the procedures of those names. Some programs however have to repeat their procedures a large number of times, the precise figure often depending on conditions arising within the data or during the calculation, and hence not known in advance. In order to deal with these requirements, a programming device known as the loop exists in almost all languages.

The function of the loop is to cause the execution of certain lines of code (the *body)* a certain number of times. Different types of loop may be distinguished by the way in which they decide how many repetitions (or *iterations)* are required. The process of deciding whether to repeat the body of the loop one more time or to continue with the rest of the program is called a *test.* Every loop therefore consists of a body and a test and is known as a *control structure* because it causes the program control or "flow" to differ from the normal sequential execution of program statements.

The most elementary type of loop is designed to execute the body a predefined number of times. This operation is controlled by an explicit *counter* variable and the test consists of comparing the value of the counter with the known finishing value. Depending on the outcome of the test, the counter is incremented (or sometimes decremented) and the body is repeated, or else program control passes to the code immediately beyond the loop.

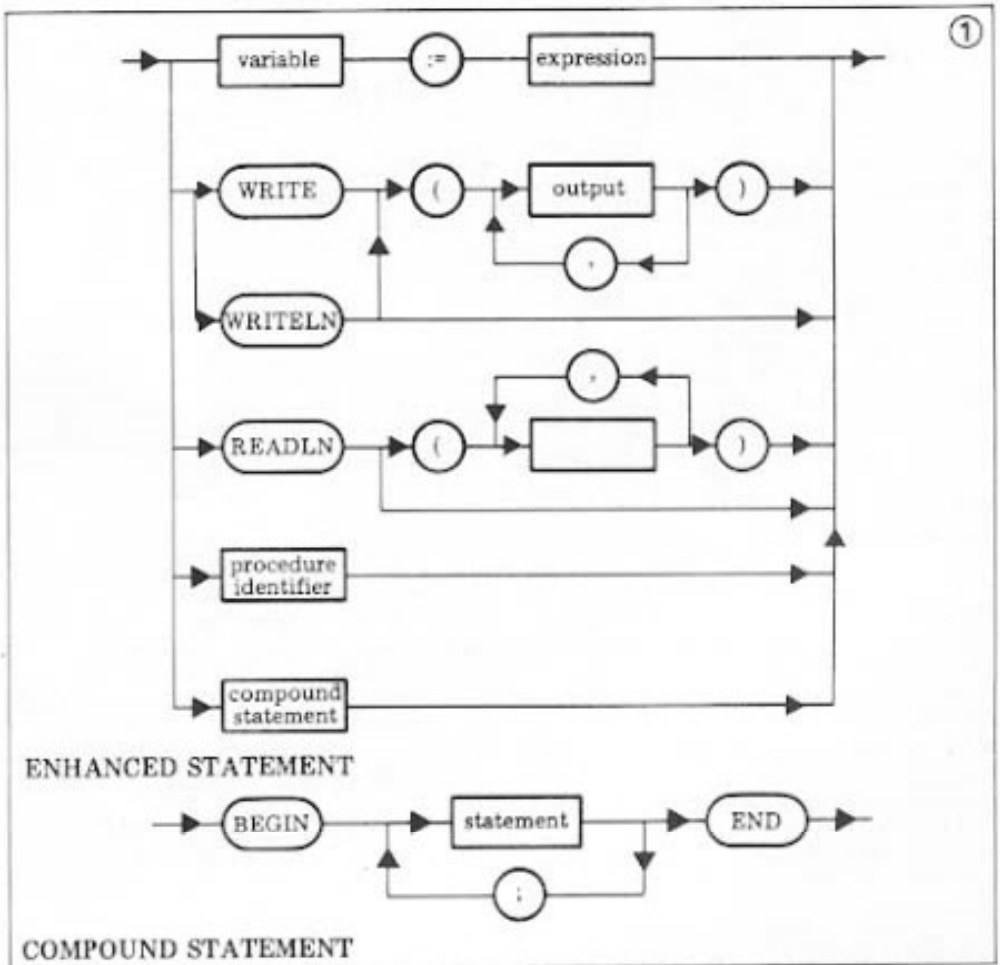
In BASIC this structure is known as a FOR-NEXT loop and PASCAL has an equivalent called the FOR-DO loop. In addition, PASCAL has two loops for executing the body an unknown (or at least uncalculated) number of times. Here the test will depend on conditions arising within the body and a counter, if used at all, is not an explicit part of the loop. In the WHILE-DO loop, the test is made before the body is commenced whereas in the REPEAT-UNTIL loop, the test comes right at the end of the body. In the next few sections each of the above will be described, defined and exemplified in programs.

The body of a loop consists of either a single statement (now expanded to include the *compound statement,* as in the syntax diagram in Box 1) or in

When laying out a program it is normal to indent the code between every BEGIN-END pair. When the body of a loop does not contain a BEGIN-END pair, however, by convention it is indented anyway, to emphasize that it is controlled within a loop.

The FOR~DO Loop

Program ROLLOVER in Box 2 illustrates a FOR-DO loop in a fairly typical context. Procedure RESTOFVERSE contains the parts of the song which are repeated in each verse. The loop, set up in line 11, ensures that the part that changes (CROWDS) is correct for each verse. This requires the special DOWNTO reserved word to make the counter work backwards. Lines 13 and 14 actually produce each verse and line 15 sends the program control back to line 11 for the next verse — and so on. Line 16 finishes off the song. Lines 12

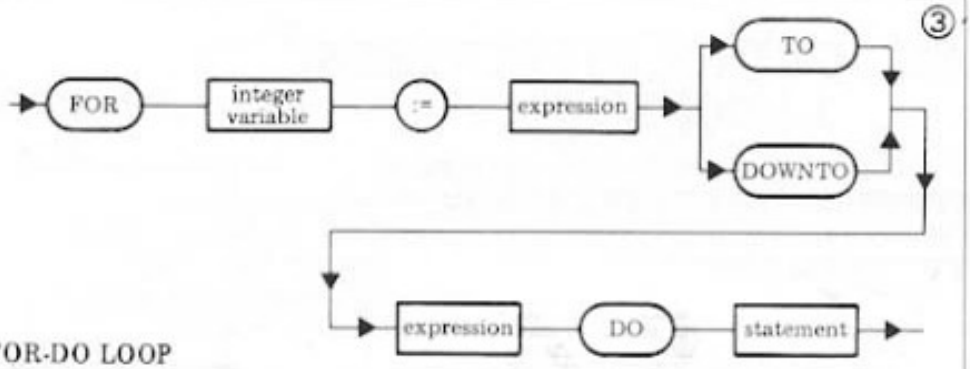


②

```
 1   PROGRAM ROLLOVER ;
 2   VAR CROWDS: INTEGER ;
 3   PROCEDURE RESTOFVERSE ;
 4   BEGIN
 5     WRITELN(' IN A BED AND THE LITTLE ONE SAID') ;
 6     WRITELN('''ROLLOVER, ROLLOVER''') ;
 7     WRITELN('SO THEY ALL ROLLED OVER AND ONE FELL OUT.') ;
 8     WRITELN
 9   END ; (*RESTOFVERSE*)
10   BEGIN (*MAIN PROGRAM*)
11     FOR CROWDS:=10 DOWNTO 2 DO
12     BEGIN
13       WRITE('THERE WERE ', CROWDS) ;
14       RESTOFVERSE
15     END ;
16     WRITELN('THERE WAS 1 IN THE BED AND HE SAID ''GOODNIGHT''.')
17   END
------------------------------------------------------------------------
18   THERE WERE 10 IN A BED AND THE LITTLE ONE SAID
19   'ROLLOVER, ROLLOVER'
20   SO THEY ALL ROLLED OVER AND ONE FELL OUT.
21
22   THERE WERE 9 IN A BED AND THE LITTLE ONE SAID
23    ...
..    ...
..    ...
54   THERE WAS ONE IN A BED AND HE SAID 'GOODNIGHT'.
```

PROGRAM ROLLOVER

③



FOR-DO LOOP

to 15 provide an example of a compound statement. Finally, note PASCAL's solution to the problem of printing a mark. Since the quote (') is the text delimiter, the PASCAL compiler searches for pairs of quotes enclosing text. Two adjacent quotes will indicate that the text is not to be terminated but rather that a single quote is required for output.

The syntax diagram in Box 3 shows the precise structure of the FOR-DO loop. The different components appear as:

FOR (test) DO (body)

The counter is a variable (*not* a REAL) and must therefore, like any other variable, be declared explicitly in the declaration part. The starting and finishing expressions must be integer expressions. Because these expressions are evaluated before the loop commences, rather than during each iteration, there is no loss of efficiency in using quite complex expressions if required.

The counter increases or decreases (depending on whether TO or DOWNTO, respectively, is used) by 1 on each iteration. The restriction of the step size creates a loop-test requiring a minimal number of machine-code instructions. If a different step size is required, a "dummy" counter can be constructed within the body of the loop, but on no account should the value of the actual counter be changed inside the loop (for obvious reasons). The FOR-DO loop test will discontinue the loop when the value of the counter moves beyond the finishing value (in the indicated direction). This ensures not only that the body is executed the correct number of times, but also, if the counter is accidently set up to move away from the finishing value, the body of the loop will be skipped over entirely.

When the loop has finished the counter variable loses any value it had is included in PASCAL as a safety measure to guard against the tendency of some programmers to re-use a loop counter at a later stage of the program, without assigning a new value to it.

EXERCISE:
Write a program to print out the song "Ten Green Bottles".

The Generalized Loop

Circumstances can often arise in programming where the use of a fixed-limit FOR-DO loop is too restrictive to allow for a fluent program style. As an example consider the problem of entering a list of numbers from a keyboard into a program. If you don't want to count how many numbers there are before you start, you need to have a way of telling the program when the list has come to an end. This is usually done with a "rogue" value — a number which couldn't possible be a part of the list (eg. -9999). When the program detects the rogue value, this is an indication that the input list is complete and further processing can continue.

It would be nice to place the item-by-item reading of such a list in a loop, but if the length of the list is unknown, then the only way of doing this with a FOR-DO loop leads to awkward and error-prone code. Because circumstances such as this arise quite frequently, PASCAL has a more generalized loop form.

The distinguishing feature of the generalized loop lies in the nature of its test. Instead of a steady incrementation of a counter, the test checks the validity of some relationship which is (presumably) affected by the body of the loop. When the relationship holds, one course of action is taken and when events within the loop cause the relationship to change, a different course of action is embarked upon. Quite clearly, only two possibilities exist — the relationship *false*). Such a relationship is called a *Boolean expression* after the English mathematician George Boole who first studied the algebra of such expressions.

The syntax diagram in Box 4 fully defines the Boolean expression. Note that <> stands for "is not equal to". Consider a Boolean expression like A=B. This expresses the relationship "A is equal to B" and the = is known as a *relational operator* as are all the other symbols shown in Box 4. Compare this with the *assignment statement* A:=B which reads "A *becomes* equal to B". Here := is an *assignment operator* and it is this distinction which enables one to write X:=X+1 in a program where it would make no sense as an equation.

PASCAL provides two versions of the generalized loop. In the first, the WHILE-DO loop, the test is made *before* the body is commenced, and iteration occurs as long as the Boolean expression is *true*. If the expression is false when the program first encounters the loop, the entire loop will be skipped. The syntax diagram in Box 5 defines a WHILE-DO loop. As with a FOR-DO loop, the body is a single statement, generally compound.

The program in Box 6 illustrates the use of a WHILE-DO loop, which runs from lines 10 to 15, line 10 containing the test and the rest comprising the body. While this is not a very practical sort of guessing game, it does show the unlimited nature of the loop which will go on asking for new guesses until the right number turns up. It also shows the major danger of the generalized loop — suppose the test never fails? The program will stay in the loop forever. For instance, suppose TARGET was 16 while CORRECT and GUESS were REAL instead of INTEGER, and CORRECT became 3.99999 (as often happens). Any integer value guessed could *never* pass the test. This can happen quite easily especially when dealing with the mathematical functions with which rounding errors are associated. Consequently, it is good programming practice to check explicitly for realisable loop tests.

Examples of mathematical functions appear in line 6. SQRT(A) is a REAL value representing $\sqrt{A}$ while TRUNC(B) is the largest integer less than B (when B is positive). In line 6 the above functions are *nested* so that CORRECT is the square-root of the largest perfect square less than TARGET. A list of all mathematical or *standard functions* available in PASCAL appears in the Look-Up Table at the end of this chapter.

The second generalized loop in PASCAL is the REPEAT-UNTIL loop defined in Box 7. The test comes at the *end* of the body and iteration occurs as long as the condition is *false*. PASCAL has two complementary loops to allow for a fluent programming style. Sometimes it will seem more natural to use a WHILE-DO loop and sometimes a REPEAT-UNTIL will suggest itself. In the latter case however, the body will be executed at least once, whatever state the Boolean expression is in, because the test comes after the body. Program ANOTHERGO in Box 8 illustrates the use of a REPEAT-UNTIL loop running from lines 22 to 26. Line 26 contains the test and the body lies above it.

EXTRA

GH • PRICE BREAKTHROUGH • PRICE BREAKTHR

3M SCOTCH 5 1/4" OR 8" DISKETTES

500

$4·50 each

100

$5·50 each

10

$6·00 each

*SUPPLIED IN SOFT CARTONS, FOR PLASTIC LIBRARY CASES ADD 0.40c EACH.

DON'T RISK YOUR DATA
USE THE BEST!

MAIL ORDER ONLY – CASH WITH ORDER – SCHOOL OR GOVT. DEPT. ACCOUNTS O.K. – ADD $5.00 PER ORDER FOR REGISTERED DELIVERY.

ATTENTION

PROGRAMMERS
MANUFACTURERS
INVENTORS

MY NAME IS
PETER DE FOREST

AND MY COMPANY
DEFOREST SOFTWARE

URGENTLY *REQUIRES GOOD PRODUCTS/PROGRAMMES FOR THE MICROCOMPUTER INDUSTRY. THIS INDUSTRY, WITH ONE OF THE HIGHEST GROWTH RATES, IS SORELY LACKING IN* **GOOD** *LOCALLY WRITTEN SOFTWARE AND INNOVATIVE LOCALLY MANUFACTURED HARDWARE.*

LET **DEFOREST SOFTWARE**
MARKET YOUR PRODUCT!

YOU CAN RING ME ON

(03) 877 6946 *OR* **(03) 878 9276**

OR SUBMIT YOUR PROGRAMMES OR IDEAS

TO ME AT

26 STATION STREET, NUNAWADING

STRICT CONFIDENCE IS ASSURED.

DEFOREST SOFTWARE

New DOS 80 – The latest and greatest from Apparat, variable length fields to 4K, mix and match drives on 1 cable. 18 to 80 track. Security boot and much more . . . $149.50

New DOS Plus – A very powerful disk operating system, too many features to list. Send 0.50c for reviews and information . . . . . 40 track $110.00
35 track $99.00

New DOS – A scaled down version of the above
40 track $59.00
35 track $49.00

Editor Assembler Plus – Microsoft – Very powerful complete with de-bugger. $34.00

Basic Compiler – Create fast Z80 code your basic programs $202.00

Level 3 Basic – Get all the advantages of disk basic plus many extra features(disk or cassette). $52.00

Assembly Language Package – Microsoft $102.00

Fortran – Microsoft – Fully compatable with TRS-DOS $102.00

Packer – Comes with 16K, 32K & 48K versions; perhaps the best program written in 1979; remove rems & spaces, renumber, move and more – Disk add $5.00. $30.00

Clone 2 – Copies system or data tapes, make backups of your valuable programs. $18.00

Print to L. Print – No Printer? Simply use this program to change all L. Prints to Prints or vice versa, other features. $10.00

System Savers – FLEX! and T DISK put your favorite M/L tape to disk or make back-ups of system tapes. $24.00

Utility 1 & 2 – from Instant Software – 2 tapes each $10.00

EDUCATION

SBT – Structure Basic Translator – write basic in a structured format . . . . . disk $23.00

Teacher – Create your own texts store on data file $12.50

French/Italian/German – Each on disk, very powerful language CAI program. . . . . each $23.00

HARDWARE

Speed up now up to 100% faster. Make your machine the fastest on the block $35.00

Reverse Video – Gives crisp black on white $31.00

Light pens . . . . . Assembled $$19.00
Kit $14.00

Flow Chart Pads. . . . . 2 Pads $8.50
5 Pads $37.50

Giant Video Display Pads . . . . . 5 Pads $37.50

House brand 5¼ Diskettes . . . . . 10 for $40.00

Blank Computer Cassettes . . . . . 5 for $9.00
10 for $16.00

Green Screen for TRS-80 1/8" flexi glass $35.00

Data Separators (not one shot) $46.00

Fine line graphics & lower case modification kit $150.00

OTHER PROGRAMS

The Electric Paintbrush – A graphics language, very powerful. $21.00

Stimulating Simulations – from Personal Software. . . $21.00

General Mathematics $12.50

CCA DMS – Data Base rep. 32K . . . . . Disk $95.00

MU MATH from MICROSOFT – a fantastic mathematic program . . . . . Disk $77.00

Azimuth Finder – Amateur Radio gives beam heading, call signs & distance for 100's of countries screen or printer $14.00

Ham Package – Instant Software $10.00

Music Tutor $12.00

Opera by Acorn – got to be heard to be believed. . . . $16.00

Engineers – Beam Analysis Program $25.00

Common basic programs (matches Osborne book) . . . $17.50

Adventure 9 (Ghost Town) $16.00

NEW – ASSEMBLY LANGUAGE INSTRUCTION COURSE – AUDIO CASSETTES, PROGRAM CASSETTE AND REFERENCE BOOK.

Learn Assembly Language the easy way. This course guides you smoothly through the complexity of Assembly Language without difficulty.. . . . . $70.00

AUSTRALIAN PROGRAM OF THE MONTH

Written by Paul Bennell for De Forest Software

DISK MASTER

END DISK CONFUSION, THIS PROGRAM:–

- Automatically reads disk name and directory
- Automatically renames disk on request
- Shows system files
- Shows free granules
- Stores directory within basic program if system crashes, recover with BASIC*
- Fast machine language sort
- Output to screen or printer
- Only requires 1 drive
- Works with any DOS
- No more lost files

A MUST FOR ANYONE WITH DISKS

ONLY **$24.95**

32K or 48K

BUSINESS NEWS

Microcomputers like the TRS-80 are really elegant pieces of hardware. The price is deceiving. Given the right programs, they can jump through hoops.

But finding the right program isn't all that easy. You can flip through pages of any magazine and find many ads for TRS-80 programs. Granted a good many of them are for fun and games, but you can still find quite a few offering business programs.

They aren't like these though.

Three of these are the genuine Osborne & Associates systems, originally designed for the $30,000 Wang computer. With a few minor modifications on them, they now work on a $4,000 TRS-80.

Here's what's on each disk:

THE ON-LINE, INTERACTIVE OSBORNE PROGRAMS

Accounts Payable: an invoice linked system that can calculate and print cheques, make reports and link fully to the general ledger.

Accounts Receiveable: also invoice-linked, it can keep track of billed an unbilled invoices, open and closed items and aging. It can print a statement and link to the general ledger.

General Ledger: this handles more than 1,750 transactions on 200 different accounts and keeps track of them by month, quarter, year and the previous three quarters. Available with or without Cash Journal option.

Invoicing: A completely interactive program that works with accounts receivable and will print your invoices.

AND AN ON-LINE INTERACTIVE TARANTO PROGRAM

Inventory Control: a custom-tailored program that looks after up to 1300 items – it gives an immediate readout on any item inquiry, including quantity and dollar total.

These programs are marvels of efficiency. They're fully documented, and you can buy the books locally or from me.

These programs only cost $99.95 each. (The Cash Journal option on the General Ledger adds another $50.) For that you get the disk and all the instructions you need.

We plan to turn that TRS-80 of yours into a serious computer.

Please send me the following programs at $99.95 each:

		book
Accounts Payable		
Accounts Receivable		
General Ledger (add $50 for Cash Journal)		
Invoicing		
Inventory Control		
AMOUNT ENCLOSED		

If you need the books, add $20 each.

Bankcard No._____ Expires_____

Builders Job Cost Program. Requires 3 disks, 32K, TRS-80. Keeps complete control over your costs, handles 100 jobs, estimates, sub-contractors etc.

$200

Real Estate Investment Analysis. Requires 32K, TRS-80 Disk and Printer – Use this program to predict your investment potential on flats, second homes, holiday homes, main homes etc. Gives figures up to 20 years in the future; takes into account inflation etc.

$35

Basic Compiler. TRS-80. Disk 32K. Turn your basic programs into fast Z80 code – Protect your programs.

$202

Electric Pencil. Turn your TRS-80 into a top quality word processor.

Disk $150; Cassette $100

AVAILABLE ON FLOPPY DISKS

ACCOUNTS RECEIVABLE/ ACCOUNTS PAYABLE

Now, in one package, you can have a *complete* Accounts Receivable/Accounts Payable (AR/AP) system! These programs will handle all the drudgery involved in processing AR/AP entries.

Each program is capable of handling up to 760 accounts and as many as 1500 entries per month. Should you need to handle more accounts, you can divide them into as many sub-groups as necessary and keep a set of data disks for each sub-group.

The Accounts Receivable program can print invoices, statements, and address labels for each customer. It will also generate a Month End report, a Customer Activity report, and a Daily Sales report.

The Accounts Payable program will generate a Month End report, an Account report, a Daily Activity report and a Check Payment report.

The AR/AP package is ideal for any small business. The programs are self-prompting and are easily used by anyone familiar with AR/AP operations.

These programs can save you money, because they can print your company's letterhead at the top of each invoice and statement, using plain, fan-folded paper.

Accounts Receivable/Accounts Payable: Software for the Professional.

THIS PACKAGE REQUIRES THE FOLLOWING MINIMUM SYSTEM

1. A TRS-80 Level II microcomputer, 16K of memory.
2. An Expansion Interface with at least 16K of additional memory.
3. Three mini-disk drives.
4. A pin-feed line printer.
5. Any TRS-80 Disk Operating System.

$210.00

bracket the body of the loop. This is not the case with the other two loops where the reserved word DO merely leads up to the beginning of the body. The PASCAL compiler needs to know where the loop body stops and the rest of the program begins. It is for this reason that the two DO loops restrict the programmer to a body consisting of a single statement (usually compound). Without the DO keyword possessed by the other loops, the REPEAT-UNTIL loop can contain more than one statement in its body (cf. syntax diagrams for the different loops). This means that one tends not to find BEGIN-END pairs following a REPEAT although the indentation convention is observed nonetheless.

The program from Box 6 has been converted into a procedure for ANOTHERGO. This is a sensible way to develop programs — writing a small, self-contained section as a separate program, testing it, and then incorporating it as a procedure in some larger program. This theme will be developed in more detail in the next section. Finally, line 2 introduces a new *data type*, the character type CHAR which consists of a single letter of the alphabet, digit or normal keyboard punctuation mark. The variable ANSWER can contain any one of these characters and can be compared with actual characters enclosed in 'quotes' as in line 26. Variables therefore can be declared as INTEGER, REAL or CHAR.

Each of the three control structures defined above is an extension of the definition of a statement, since it appears in the action part of a program. Consequently a complete syntax diagram for the statement must incorporate all of these, and this is shown in Box 9.

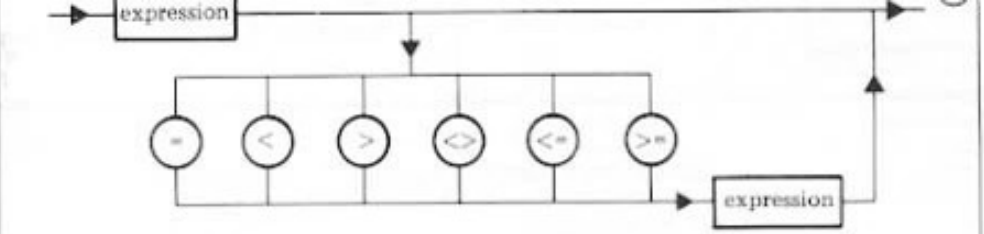
EXERCISE:
Computers (and calculators) are often tested for accuracy by computing a range of nested mutually inverse functions [eg. exp (ln[x]) = x].
Write a program to input a sequence of (positive) numbers (rogue values could be 0 or less), in each case calculating exp (ln[x]) and outputting this value, together with x and the difference between them before reading in the next one.

Using Loops

As an everyday application of the use of loops, consider the construction of a mortgage repayment table. These are normally constructed by actuaries from formulae which give the monthly payment incurred by a loan assuming a fixed interest rate and where repayment occurs over a fixed time period.

This reputedly boring occupation seems ideally suited for rendering into machine soluble form, releasing the actuary for more valuable tasks (like estimating the insurance risks on a personal computer). Instead of employing the actuarial formula, however, the problem will be used to illustrate a common programming technique which consists of taking a guess at the likely value, working out the implications, comparing the results with the required outcome, improving the guess, working out the implications again, and repeating this process until an acceptable answer is reached. Clearly, the loop provides a means of programming such an iterative solution — although it's unlikely to tempt any actuaries away from their formulae!

The approach we shall take in programming this problem is known as "Top-Down Design". The Top-Down designer begins by explicitly defining the problem, stating what results are expected from what initial information. The task is then coded by calling several procedures, each a distinct subtask or *module* which contributes to the solution of the total problem. Any consideration of the detail of these



BOOLEAN EXPRESSION

⑤

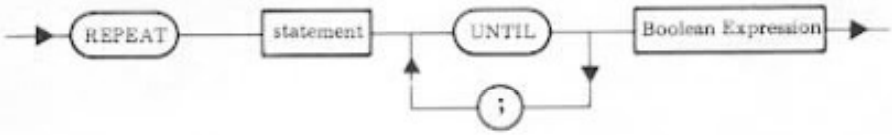
WHILE-DO LOOP

⑥

```
 1   PROGRAM PERFECTSQUARE ;
 2   VAR CORRECT, GUESS, TARGET: INTEGER ;
 3   BEGIN
 4     WRITE('TYPE IN YOUR TARGET NUMBER:-') ;
 5     READLN(TARGET) ;
 6     CORRECT:=TRUNC(SQRT(TARGET)) ;
 7     WRITE('NOW GUESS THE LARGEST INTEGER YOU THINK HAS A SQUARE ',
 8           'NOT LARGER THAN ', TARGET, '!-') ;
 9     READLN(GUESS) ;
10     WHILE GUESS<>CORRECT DO
11     BEGIN
12       WRITELN('NO THAT GIVES ', GUESS*GUESS) ;
13       WRITE('SO GUESS AGAIN!- ') ;
14       READLN(GUESS)
15     END ;
16     WRITELN('GOOD ', GUESS, ' HAS THE LARGEST PERFECT SQUARE ',
17             ' NOT LARGER THAN ', TARGET,'.')
18   END . {*PERFECTSQUARE*}
-------------------------------------------------------------------
19   TYPE IN YOUR TARGET NUMBER:- 59
20   NOW GUESS THE LARGEST INTEGER YOU THINK HAS A SQUARE NOT LARGER THAN 59!- 6
21   NO THAT GIVES   36
22   SO GUESS AGAIN!- 7
23   GOOD    7 HAS THE LARGEST PERFECT SQUARE NOT LARGER THAN 59.
```

PROGRAM PERFECT SQUARE

⑦

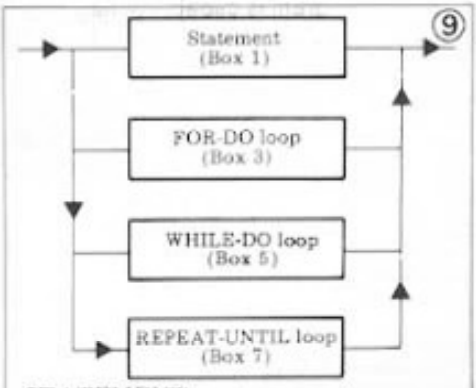


REPEAT-UNTIL LOOP

PROGRAM ANOTHERGO ⑧

```
 1   PROGRAM ANOTHERGO ;
 2   VAR ANSWER: CHAR ;
 3   PROCEDURE PERFECTSQUARE ;
 4   VAR CORRECT, GUESS, TARGET: INTEGER ;
 5   BEGIN
 6     WRITE('TYPE IN YOUR TARGET NUMBER:-') ;
 7     READLN(TARGET) ;
 8     CORRECT:=TRUNC(SQRT(TARGET)) ;
 9     WRITE('NOW GUESS THE LARGEST INTEGER YOU THINK HAS A SQUARE ',
10           'NOT LARGER THAN ', TARGET, '!-') ;
11     READLN(GUESS) ;
12     WHILE GUESS<>CORRECT DO
13     BEGIN
14       WRITELN('NO THAT GIVES ', GUESS*GUESS) ;
15       WRITE('SO GUESS AGAIN!- ') ;
16       READLN(GUESS)
17     END ;
18     WRITELN('GOOD ', GUESS, ' HAS THE LARGEST PERFECT SQUARE ',
19             ' NOT LARGER THAN ', TARGET,'.')
20   END ; {*PERFECTSQUARE*}
21   BEGIN {*MAIN PROGRAM*}
22     REPEAT
23       PERFECTSQUARE ;
24       WRITE('DO YOU WANT TO TRY ANOTHER TARGET?  ') ;
25       READLN(ANSWER)
26     UNTIL ANSWER='N'
27   END.
-------------------------------------------------------------------
28   TYPE IN YOUR TARGET NUMBER:- 44
29   NOW GUESS THE LARGEST INTEGER YOU THINK HAS A SQUARE NOT LARGER THAN 44!-
30   GOOD    6 HAS THE LARGEST PERFECT SQUARE NOT LARGER THAN  44.
31   DO YOU WANT TO TRY ANOTHER TARGET? N
```

⑨



STATEMENT

Cottage Computers

a division of Embryonic Systems Pty. Ltd.

We are the southern agents for MicroByte

MicroByte

Software for the 2650

This fully tested, well documented software will turn your 2650 based system into a powerful, flexible microcomputer . . . . . .
WE STOCK ALL ITEMS HERE at Cottage Computers

ACOS
BINBUG 4.4 (VDU)
BINBUG 5.2 (Terminal)

ASSEMBLER
BASIC
SOURCE GENERATOR

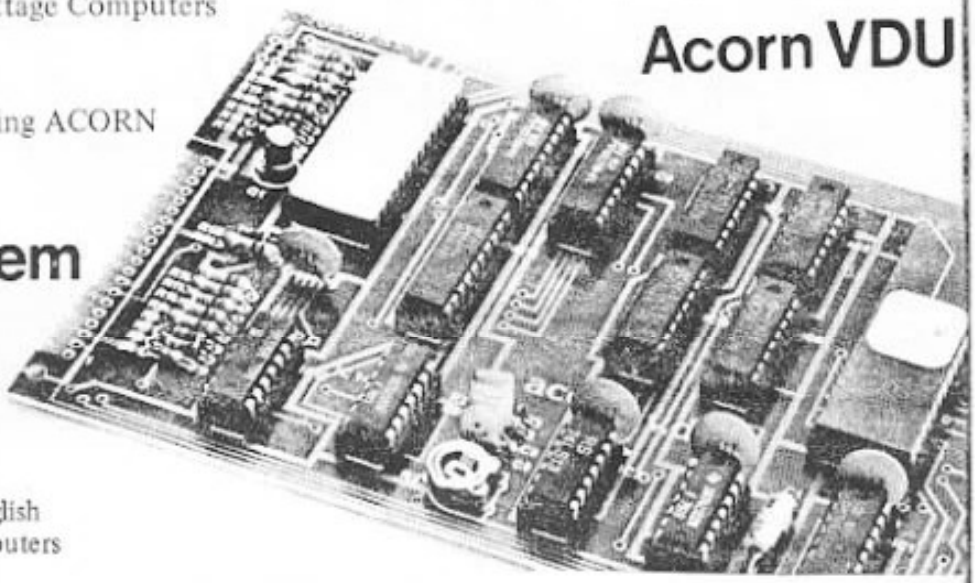
Full Product Info Sheets available

HERE IN AUSTRALIA!!!!The Amazing ACORN

Acorn System

6502 or 6809 MPU
EUROCARD Modules
TELETEXT Compatible
Colour VDU Card
ACORN COS or DOS
4K BASIC in ROM
Come and see this remarkable english computer in action at cottage computers

Acorn VDU



Science of Cambridge

The MK14 is a kit of parts assembled by the user to form a minimum cost computer. It comprises in miniature the essential elements common to all computer systems.

**MK 14 Kit	$93
**Cassette I/face	$14
**VDU Kit	$69
**PROM Prog.	$24

Extra Programs available soon

- **Central Processor Unit** SCMP CPU
- **Fixed Memory** 512 bytes
- **Read/Write Memory** 256 bytes
- **Users Terminal**
- **External Input/Output** Edge connection for 16 I/O ports, 8 data bus, flags and sense lines and supply

bankcard welcome here

Telephone: (03) 481 1975

```
2   VAR MIN, MAX, LOAN, REPAY: INTEGER ;
3     PROCEDURE GETINPUTS ;
4               (*READ IN INTEREST RATE, NUMBER OF YEARS,
5                 MINIMUM AND MAXIMUM LOANS*)
6     PROCEDURE PRINTHEADINGS ;
7               (*PRINT OUT INTEREST RATE, NUMBER OF YEARS
8                 AND TABLE HEADINGS-I.E.LOAN & REPAYMENTS*)
9     PROCEDURE CALCULATEREPAY ;
10              (*WORK OUT MONTHLY REPAYMENTS*)
11    BEGIN (*MAIN PROGRAM*)
12      GETINPUTS ;
13      PRINTHEADING ;
14      LOAN:=MIN ;
15      WHILE LOAN<=MAX DO
16      BEGIN
17        CALCULATEREPAY ;
18        WRITELN(LOAN, '     ', REPAY) ;
19        LOAN:=LOAN + 1000
20      END
21    END. (*REPAYMENTS*)
```

PROGRAM REPAYMENTS — FIRST ATTEMPT

(11)

```
1   PROCEDURE CALCULATEREPAY ;
2   VAR TOTALMONTHS: INTEGER ;
3       MONTHLYINTERESTRATE, AMOUNTDUE: REAL ;
4   PROCEDURE TRYREPAY ;
5             (*WORK OUT THE ACTUAL AMOUNT A GIVEN REPAYMENT
6               WILL ACTUALLY PAY OFF*)
7   BEGIN
8     MONTHLYINTERESTRATE:=INTERESTRATE/12 ;
9     TOTALMONTHS:=12*YEARS ;
10    REPAY:=LOAN DIV TOTALMONTHS ;
11    REPEAT
12      AMOUNTDUE:=LOAN ;
13      REPAY:=REPAY + 1 ;
14      TRYREPAY
15    UNTIL AMOUNTDUE<=0
16  END ; (*CALCULATEREPAY*)
```

PROCEDURE CALCULATEREPAY

(12)

```
1   PROCEDURE TRYREPAY ;
2             (*WORK OUT THE ACTUAL AMOUNT A GIVEN REPAYMENT
3               WILL ACTUALLY PAY OFF*)
4   VAR MONTH:INTEGER ;
5   BEGIN (*CALCULATEREPAY*)
6     FOR MONTH:=1 TO TOTALMONTHS DO
7       AMOUNTDUE:=(AMOUNTDUE-REPAY)*(1 + MONTHLYINTERESTRATE)
8   END ; (*TRYREPAY*)
```

PROCEDURE TRYREPAY

(13)

```
1   PROCEDURE GETINPUTS ;
2   CONST IMIN=2 ; IMAX=50 ;
3         YMIN=5 ; YMAX=35 ;
4         LMIN=5 ; LMAX=200 ;
5   PROCEDURE GETINTEREST ;
6             (*READS IN INTEREST RATE BETWEEN IMIN AND IMAX AND
7               CONVERTS IT TO A DECIMAL*)
8   PROCEDURE GETYEARS ;
9             (*READS IN DURATION OF LOAN BETWEEN YMIN AND YMAX YEARS*)
10  PROCEDURE GETMIN ;
11            (*READS IN, IN THOUSANDS, THE MINIMUM LOAN VALUE BETWEEN
12              LMIN AND LMAX AND CONVERTS IT TO DOLLARS*)
13  PROCEDURE GETMAX ;
14            (*LIKE GETMIN, BUT FOR THE MAXIMAL LOAN VALUE*)
15  BEGIN (*GETINPUTS*)
16    GETINTEREST ;
17    GETYEARS ;
18    GETMIN ;
19    GETMAX
20  END ; (*GETINPUTS*)
```

PROCEDURE GETINPUTS

(14)

```
1   PROCEDURE GETINTEREST ;
2             (*READS IN INTEREST RATE BETWEEN IMIN AND IMAX AND
3               CONVERTS IT TO A DECIMAL*)
4   BEGIN
5     WRITELN ('TYPE IN THE RATE OF INTEREST AS A PERCENTAGE.') ;
6     REPEAT
7       WRITE ('A NUMBER BETWEEN', IMIN, ' AND', IMAX, ':-') ;
8       READLN (INTERESTRATE)
9     UNTIL (INTERESTRATE>=IMIN) AND (INTERESTRATE<=IMAX) ;
10    INTERESTRATE := INTERESTRATE/100 ; (* % -> DECIMAL *)
11  END (*GETINTEREST*) ;
12  ------------------------------------------------------------
13  PROCEDURE GETYEARS ;
14            (*READS IN DURATION OF LOAN BETWEEN YMIN AND YMAX YEARS*)
15  BEGIN
16    WRITELN ('TYPE IN NUMBER OF YEARS FOR WHICH MORTGAGE WILL RUN.')
17    REPEAT
18      WRITE ('A NUMBER BETWEEN', YMIN, ' AND', YMAX, ':-') ;
19      READLN (YEARS)
20    UNTIL (YEARS>=YMIN) AND (YEARS<=YMAX)
21  END; (*GETYEARS*)
22  ------------------------------------------------------------
23  PROCEDURE GETMIN ;
24            (*READS IN, IN THOUSANDS, THE MINIMUM LOAN VALUE BETWEEN
25              LMIN AND LMAX AND CONVERTS IT TO DOLLARS*)
26  VAR LOANMIN : INTEGER ;
27  BEGIN
28    WRITELN('TYPE IN THE SMALLEST MORTGAGE YOU ARE INTERESTED IN.',
29            'IN THOUSANDS.') ;
30    REPEAT
31      WRITE ('A NUMBER BETWEEN', LMIN, ' AND', LMAX, ':-') ;
32      READLN (LOANMIN)
33    UNTIL (LOANMIN>=LMIN) AND (LOANMIN<=LMAX);
34    MIN := LOANMIN*1000
35  END (*GETMIN*) ;
36  ------------------------------------------------------------
37  PROCEDURE GETMAX ;
38            (*LIKE GETMIN, BUT FOR THE MAXIMAL LOAN VALUE*)
39  VAR LOANMAX : INTEGER ;
40  BEGIN
41    WRITELN ('TYPE IN THE LARGEST MORTGAGE YOU ARE INTERESTED IN.
42              'IN THOUSANDS.') ;
43    REPEAT
44      WRITE ('A NUMBER BETWEEN', MIN DIV 1000, ' AND', LMAX, ':-') ;
45      READLN (LOANMAX)
46    UNTIL (LOANMAX*MIN DIV 1000) AND (LOANMAX<=LMAX) ;
47    MAX := LOANMAX*1000
48  END (*GETMAX*) ;
```

PROCEDURES GETINTEREST ETC.

will undergo the same treatment and thus the problem devolves into a hierarchy of more-or-less independent subproblems until a level is reached at which only elementary programming functions are required. At this point the final coding can be done quickly and accurately, and the result should be a well-structured program.

Returning to the mortgage table program, the problem definition could be:

Given the interest rate and a time period for repayment, create a table showing the monthly payment due over a given range of loans.

The input data required is therefore:

1. interest rate (% p.a.)
2. repayment period (years)
3. maximum and minimum loans required (thousands of dollars.)

The output should be a list of loans from minimum to maximum in steps of $1000, showing monthly repayments. The interest rate and repayment period should also be displayed.

The next stage is to decide on the method of solution in order to code the main program. At this level the tasks that must be accomplished include reading in the user's parameters, printing out the appropriate headings and, for each loan from the minimum to the maximum requested, calculating and printing the repayment amount. At this stage, the means by which the calculations are to be performed do not concern us and neither are we interested in the details of getting the input data or printing out the heading. The calculations will have to be performed in a loop which will stop when the maximum loan value is reached. In Box 10, we have called procedures named GETINPUTS and PRINTHEADINGS to handle the initial part of the problem, and introduced a WHILE-DO loop (lines 15 - 20) to control the calculation and output of the table. Procedure CALCULATEREPAY will actually perform the calculations.

The declaration part of this first attempt includes all identifiers used in the main program. These include the integer variables MIN, MAX, LOAN and REPAY, together with the procedures GETINPUTS, PRINTHEADINGS and CALCULATEREPAY. Notice that these procedures have not been fully defined at this stage but merely contain a comment indicating what each will eventually do.

EXERCISE:
Try re-writing this first attempt with a FOR-DO loop instead of a WHILE-DO loop .

We have now completed the highest level of the program design and are ready to proceed to the next level. The three procedures will be tackled in the same way that the whole problem REPAYMENTS was approached. The question arises as to which of the three should be dealt with first. We prefer to start with the "Heart" of the problem — CALCULATEREPAY (Box 11). The problem definition of CALCULATEREPAY could be:

Work out the monthly repayment as follows — first guess an obviously low value and calculate how much that

C.I.S.A.
159 KENT STREET, SYDNEY

The One-stop Microcomputer Shop Providing a Total Service to TRS-80* and System 80 Users.

(TRS-80 is a registered Trademark of Tandy Radio-Shack)

HARDWARE AND MODIFICATIONS

Having system crashes?
Plug in our line filter and get main-frame stability . . . $ 55.00

Use your TRS-80 as a telephone book and automatic dialler. No hardware mods required. All software included . . . $ 59.50

LIGHT PEN with demonstration software and full programming instructions.STILL ONLY . . . $ 19.50
Lower case mod for ELECTRIC PENCIL and SCRIPSIT. Fitted and 90 day guarantee . . . $ 49.95

VIDEO STABILIZER CRYSTAL. Get professional stability on your VDU. CRYSTAL only . . . $ 19.95
Fitted and guaranteed . . . $ 39.00

16K MEMORY KITS . . . $ 85.00
We install . . . $105.00

RS-232 PRINTER DRIVER. Operates from cassette cable. Including software . . . $ 54.75

CISA DATA DIGITIZER. Our most popular hardware product. No hardware mods required. Reads and re-records tapes normally unreadable by any other method . . . $ 57.50

CASSETTE CABINET. Protect your valuable cassettes in this strong, handsome 36 cassette cabinet . . . $ 27.75

DIGITAL CASSETTES – Top quality

C15's	1 – $1.65	10 – $14.85
C20's	1 – $1.85	10 – $16.85
C30's	1 – $1.99	10 – $18.50

This is an entirely new product. We are so confident of this product that we are now recording all our commercial software on it.

5¼" PRIME QUALITY DISKETTES

1 – $4.95 10 – $45.00

All tracks guaranteed to load

We are extending our range of computer paper and labels for the enthusiast and the business user.

LATE SPECIAL!

EXATRON STRINGY FLOPPIES . . . P.O.A.
77 Track Micropolis Drives complete with DOS . . . P.O.A.

BOOKS

We believe that we have the best and most comprehensive range of titles of interest to the TRS-80 owner/user. Over 200 titles presently in stock. COME IN AND BROWSE AROUND!

JUST A SELECTION

First Book of Microcomputers	$ 6.50
Pascal with Style	$ 8.50
Basic with Style	$ 8.00
TRS-80 Disk and Other Mysteries	$29.50
Level II ROM Reference Manual	$24.95
Basic and the Personal Computer (highly recommended)	$15.95
Basic Programming for Scientists and Engineers	$15.50
The Incredible Secret Money Machine	$ 7.95
Z-80 Microcomputer Handbook	$11.95
Microcomputers for Business Applications	$11.95
Z-80 and 8080 Assembly Language Programming	$10.00
How to Profit from your Personal Computer	$12.00
Little Book of Basic Style	$ 7.95
Microcomputer Interfacing	$25.75
Microcomputers and the 3 R's (A Guide for Teachers)	$10.00

COMPUTER CLASSICS

Wirth/Algorithms + Data Structures = Programs	$29.75
Tausworth/Standardized Development of Computer Software	
Vol. I – Methods	$29.75
Vol. II – Standards	$29.75
Klingman/Microprocessor Systems Design	$30.50
Gilmour/Business Systems Handbook	$29.75
Gane/Structured Systems Analysis	$26.50
Garrett/Analog Systems for Microprocessors	$26.50
Rhyne/Fundamentals of Digital Systems Design	$26.95

TRS-80 AND SYSTEM 80 SOFTWARE

Ecology Simulations 1 and 2 (disk and lower case) each	$24.95
Mission Impossible and Voodoo Castle (disk)	$24.95
Adventureland and Pirate Adventure (disk)	$24.95
Investment Analysis (disk or L2 cassette)	$49.95
Sargon II (cassette)	$37.50
Strategy Games (5) (cassette)	$ 7.95
Space Games (4) (cassette)	$ 7.95
5 Adventures (cassette) each	$14.95
Advanced Statistics (disk or cassette)	$24.95
U Boat (cassette)	$ 8.50
Basic Monitor (cassette)	$19.95
Micromazing (games cassette)	$ 7.50
RPN Calculator (cassette)	$14.95

PLUS MANY MORE

We stock and highly recommend all RACET COMPUTES professional standard system programs.

Infinite Basic	$ 49.95
Infinite Business	$ 29.95
Remodel	$ 24.95
Remodel + Proloud	$ 34.95
GSF	$ 24.95
DOSORT (sorts disk files)	$ 34.95
COPSYS (copies system tapes)	$ 34.95

NEW DOS FROM APPARAT

35 Track	$ 99.00
40 Track	$110.00

This DOS emulates many mainframe capabilities. Has features usually restricted to much larger systems.

TARANTO and ASSOCIATES

General Ledger	$155.00
Accounts Payable	$155.00
Accounts Receivable	$155.00
Print Invoice	$ 99.00
Complete Business System	$500.00

This is professional quality software and really provides a superb business system for the TRS-80. Software maintenance is available to purchasers of the complete system at $15 per hour, to tailor the programs to your specific needs.

PLUS MANY MORE

We are the Australian agents for many first class U.S. software houses. Send for our full catalogue.

Remember, for all your TRS-80 and System 80 needs, phone, write or call

TRADE ENQUIRIES INVITED

Showroom:
Ground Floor, GIO Building, 159 Kent Street, Sydney 2000
(opposite IBM Building)
Phone: (02) 241 1813

We regularly conduct a series of evening periods of small group tuition from 7.30 p.m. onwards in BASIC with specific emphasis on commercial applications.
These sessions are conducted in a friendly and helpful atmosphere and are designed to give you a complete mastery of BASIC techniques up to and including all aspects of disc I/O and file handling.
We will teach you tricks even Tandy don't know about!
All stationery and workbooks are provided. Textbooks available at an extra charge.
Fee for 10 periods (up to 3 hours each) $150.00
Special private or group tuition on any aspect of BASIC or commercial/business applications of microcomputers available on request.
Tuition includes assignment work if required – and assessment.
A certificate of competence is awarded on successful completion of a tuition series.

REMEMBER – for all your TRS-80 needs – phone, write or call

159 KENT STREET,
SYDNEY 2000
(opposite IBM building)
Phone: (02) 241 1813

We are currently preparing a correspondence course for out-of-town TRS-80 users. Details will be available shortly.

We carry the full range of MICRO-80 newsletters and recorded software plus a range of popular micro-computing journals.

would pay off over the given time period, taking into account the interest charges. If there is still a debt by the end, the repayment value was not enough, so increase it and try again. Continue until the repayment amount pays off the loan.
Input data
1. duration of loan
2. interest rate
3. amount of loan
Output data is the calculated monthly repayment amount.

In the declaration part, the variables required in the calculation will have to be declared only if they are local to the procedure, since the global variables will already have been declared. Thus a check should be made that the input and output variables, YEARS, INTERESTRATE, LOAN and REPAY appear in the variable declaration of the main program. Some of these may be missing in a "first attempt" version and so should be incorporated.

To start coding CALCULATEREPAY the first step is to generate the working data from the input data. The repayment period, for instance, is in years but is here required in months as is the interest rate. Therefore two new (local) variables TOTALMONTHS and MONTHLYINTERESTRATE must be declared and calculated. Next, the initial estimate should be made, in order to start the whole process off. Since repayments will be *increased* to improve the "guess", it is important to start with an estimate *below* the likely value. A reasonable first estimate would be the amount one would pay back interest-free. This is simple enough to code at this stage as can be seen in line 10 of Box 11. (Note that DIV has been used since REPAY is an integer. This program could be changed to give dollars and cents if the user were willing to trade some speed for such accuracy). Since the initial estimate must be too low, the next step should be to add $1 to the repayment and test whether that will pay off the loan.

The process of incrementing the repayment amount and testing will be repeated until a figure is reached which actually *does* pay off the loan. This has been coded in the REPEAT-UNTIL loop, lines 11 to 15, Box 11, but, just as this calculation was put off in the main program, so the job of calculating how much a given value of REPAY would actually pay off over the time-period is deferred to procedure TRYREPAY, which is the next problem to be tackled (Box 12).

The problem definition of TRYREPAY could be:
Evaluate how much a given value of REPAY would pay off over the given duration of the mortgage using the given interest rate, assuming monthly payments and the compounding of interest.
Input data
1. monthly interest rate
2. duration of loan (months)
3. value of loan ($)
4. value of repayment ($ per month)
Output data — amount of debt remaining when time period has elapsed.

What is owing at the end of one month? Suppose AMOUNTDUE contains the amount due at the beginning of one month and an amount REPAY is paid back. At the end of that month, the amount due will be (AMOUNTDUE - REPAY) + interest accrued during the month. This figure will become the AMOUNTDUE for the next month; for N months, this calculation should pass through N iterations.

This is coded in the FOR-DO loop, Box 12, lines 6 and 7. The only variable needed that has not been previously declared is the loop counter, which is declared locally in line 4. This completes the definition of TRYREPAY which, in turn, completes the definition of procedure CALCULATEREPAY.

Having coded CALCULATEREPAY we now know exactly what information GETINPUTS must obtain. The problem definition could be:

Read in interest rate, duration of loan and maximum and minimum loans (in thousands of dollars). Convert interest rate to a decimal (instead of percentage) and loan values to dollars.
Output data
1. interest rate (decimal fraction)
2. duration of loan
3. minimum loan
4. maximum loan

An input procedure should usually check that the data it accepts is reasonable and unlikely to cause the program to crash. For instance, if the repayment period YEARS were zero, then TOTALMONTHS would also be zero. But we divide by TOTALMONTHS in CALCULATEREPAY, so that apart from zero being an unreasonable figure for years it will also crash the program.

Box 13 contains procedure GETINPUTS. In the action part the four procedures GETINTEREST, GETYEARS, GETMIN and GETMAX are called. The declaration part lays down limits within which the input data should fall (lines 2 - 4). If one of these should later on prove restricting, it will be easy to change the CONST declaration.

The four individual input procedures (Box 14) are so similar that only one, GETINTEREST, need be considered in detail. Its problem definition could be:

Output a message asking for the rate of interest. Check whether the response is within the range of reasonable values. Keep asking until an acceptable reply is received. Then convert this number from a percentage to a decimal fraction.
Input Data
IMIN and IMAX — limits of "reasonable" interest rates (as a percentage).
Output Data
INTERESTRATE — actual required interest rate as a decimal fraction.

A REPEAT-UNTIL loop (lines 6 to 9) is used to accept input. The program remains in the loop until an acceptable figure is entered.

The other three input procedures are developed in a similar fashion. Note that in procedure GETMAX, the minimum value for a loan is not LMIN but MIN DIV 1000 — the actual lower limit obtained from GETMIN (line 34).

Finally, PRINTHEADING is tackled (Box 15). Its problem definition could be:

Clear the screen, then print out a title followed by the required interest-rate and the duration of the loan. Skip several lines and print the headings MORTGAGE (for the loan) and MONTHLY REPAYMENTS.
Input Data
1. yearly interest rate (%)
2. duration of loan (years)
Output Data — none as this procedure simply produces the headings.

Cont. on page 38.

(15)

```
1  PROCEDURE PRINTHEADINGS ;
2            (*PRINT OUT INTEREST RATE, NUMBER OF YEARS AND
3              TABLE HEADINGS-I.E. LOAN AND REPAYMENTS*)
4  CONST SPACE='                    ' ;
5  BEGIN
6    WRITELN(SPACE, '**MONTHLY MORTGAGE REPAYMENTS**') ;
7    WRITELN(SPACE, '---------- -------- ----------') ;
8    WRITELN ;
9    WRITELN('INTEREST RATE =', 100*INTERESTRATE, '% OVER ',
           YEARS, 'YEARS') ;
10   WRITELN('    LOAN      REPAYMENTS') ;
11   WRITELN('    ----      ----------') ;
12   WRITELN
13 END ; (*PRINTHEADINGS*)
```

PROCEDURE PRINTHEADINGS

Look up table

PASCAL STANDARD FUNCTIONS

Arithmetic:		
ABS(X)	Absolute Value	Real or Integer
SIN (X)	Trig functions	Answer Real
COS (X)		
ARCTAN (X)		
EXP (X)	Exponential	Answer Real
LN (X)	Natural Logs	
SQR (X)	Square	Real or Integer
SQRT (X)	Square Root	Answer Real
Transfer:		
TRUNC(X)	Truncate	X real, Answer is integer part
ROUND(X)	Round to closest integer	

[ROUND(X) = TRUNC(X + 0.5) when X is positive]

COMPUTING JARGON

Control Structure
Counter
Compound Statement
Rogue Value
Boolean Expression
Relational Operator
Module
Iteration

PASCAL RESERVED WORDS

FOR
DO
TO
DOWNTO
WHILE
REPEAT
UNTIL
CHAR

UCSD Exceptions

ATAN(X) instead of ARCTAN(X)
Also LOG(X) is log to base 10.

EXERCISE SUMMARY

1. Ten Green Bottles
2. Accuracy Test
3. Mortgage Table

CHECKOUT

EXATRON STRINGY FLOPPY

Thomas Murphy gives his personal impressions of a device that looks set to radicalize the concept of information storage for the small computer enthusiast.

Whilst waiting for my SWTPC 6800 to load 8K BASIC via its 300 baud cassette interface (some 14+ minutes worth), I happened to browse through an American computer magazine and spotted an advertisement by the Exatron Corporation for their Stringy Floppy; it was claimed that the combination gave: (a) economy of tape, with (b) the speed and reliability of discs. Apparently, this system reads and writes at 14,500 bits per second, with a typical error rate of 1 in 100,000,000 bits. They also claim an average life of over 3500 hours for the transport mechanism, and a tape wafer life of 2500 passes.

My BASIC was barely half loaded, so I filled in time by writing to them at: 3557 Ryder Street, Santa Clara, California 95051, USA. I explained that I was VERY interested in their system for my SWTPC computer. Less than two weeks later a large envelope arrived, containing total system information and advising that payment could be made by the indicated method.

Could all the claims contained within this information package be true at such an attractive price? Well one way to find out was to place an order – which I duly did – for one drive mechanism and a controller card, with TSC BASIC as an extra piece of software. I felt I couldn't lost. With a 30 day money back guarantee, all it would cost would be postage and packing charges should the equipment not perform as advertised.

Five weeks later, the parcel arrived, containing all I had ordered, plus a couple of spare tape wafers, and two guaranteed system master wafers.

I settled into my favourite armchair to study the large owners' manual (which, amongst other things, tells you that Exatron's logo, of an E inside another E, stands for excellence in electronics). It proved very comprehensive, and surprise number one, the systems water contains SWTPC's Disc BASIC – free – as well as the ordered TSC BASIC.

The manual also contains a system description, system requirements, installation and checkout procedure (which includes trouble-shooting procedures for both the controller and transport), circuit and block diagrams of both controller and transport electronics, a general guide to system operation, and a detailed overview of each utility program on the systems wafer.

These programs are: APPEND, ASSIGN, CATALOGUE, COPY, DATE, DELETE, LOST, NEWTAPE, PRINT (which causes the file to be output to the printer on PORT 7 instead of the VDU), RENAME, SAVE, SAVE LOW, TTYSET (with which one can change the input and output parameters to the terminal), and VERSION. There is no LOAD, as you call a file by giving file details – e.g. 1. STARTREK. HEX will load STARTREK from drive number one.

The manual also contains an ERROR LIST for the system, plus the manufacturer's User's Manual for SWTPC Disc BASIC, plus any optional software ordered.

After reading the manual twice, I installed the controller on the motherboard,plugged in the transport,switched on and typed Z which on my monitor executes a jump to $C000. The transport started running, stopped, and instead of outputting "Simplex –68 Version X.X" as called for in the manual, my micro returned to its reset state. Oh well, back to the drawing board . . . I re-read the manual. Finding there was nothing that I had done wrongly, I typed Z again. Same result, back to monitor.

OK, call up the heavy artillery . . . out with the oscilloscope. I went

Facing page: The stringy floppy drive. Above: Inside the wafer, showing the tape being drawn from the centre of the spool and rewound on the outside.

through the troubleshooting part of the manual with probe in hand. Everything checked out, so I should have had a plus and minus 25% speed variation on the drive. I tried again, but with the same result.

Sending it back seemed the only solution – and in the course of unplugging the drive, the ribbon cable came away instead of the plug! In my excitement (it was 3 am) I think I reconnected it back to front, because on the next try, although the drive motor came on, it wouldn't even switch off and go back to monitor.

I parcelled it all up and returned it for inspection/repair with a letter explaining how I thought I had abused it. I also ordered the second drive unit at the same time; now I had the owners' manual I could see the versatility offered by having two drives instead of one.

The manual says that Exatron will repair "within 30 days", so allowing for airmail both ways, I settled back for about an eight week wait.

Surprise number two: 23 days later my system reappeared, complete with the second drive as ordered, a couple more spare wafers, and yet another very pleasant surprise – the repair charge was NIL. That's right, absolutely NO CHARGE, although I know there were some chips burnt as a result of my tiredness (carelessness).

As I now knew the owners' manual almost by heart, I installed the system in the micro, connected up, inserted a wafer in each drive, and typed Z. The drives switched on consecutively, and up came "SIMPLEX – 68 Version 1.0" on the VDU.

It took a few moments of gloating to realise that I had around 80K (well, two identical lots of 40K) of software just waiting to be called up. Just think, though, no worry about volume control, tone control, different cassettes, each with two sides, or, where on tape was the wanted file. . . just call for what you want. MAGICAL.

I typed CAT,0,1 and 30 seconds later I had seen, on my VDU the directory of both drives.

Before you can WRITE a file, you must use the NEWTAPE facility, so this I now did, and after the prompt "SCRATCH TAPE IN DRIVE 1" was answered with a "Y", (after replacing the backup system master wafer with a new 50 foot wafer in drive 1) the drive started up, stopped, and the message "FORMATTING COMPLETE – 318 SECTORS FREE" came up. Each sector holds 256 bytes, so my newly formatted wafer would hold 79.5 Kbytes.

I decided to transfer my BASIC library from cassette, and as SWTPC Disc BASIC supports a "tape load" (TLOAD) command, to pull data from cassette, I typed "BASIC". The drive searches at 20 inches per second, and reads/writes at 10 inches per second; around 30 seconds later the VDU showed "READY".

The wafer is a small (1.6 x 2.7 x 0.2 of an inch) cartridge and the length of tape can be 5, 10, 20 or 50 feet; it's of the endless loop variety, i.e. like the car 8-track cartridge and you can dramatically improve access time at the expense of the amount of data stored on the wafer. After two years plus of 300 baud cassette operation I am quite happy to wait the 30 or maybe 40 seconds (worst case) for 10K of BASIC to be loaded and executed from the longest wafer.

By careful arrangement of my games tape, I can be playing my own version of Startrek (some 5K long) within 15 seconds of initialising the system; it's the most popular household game and it's first on the wafer!

This file has memory requirements from $0000 to $13EF, plus a random number generator located at $A04A to $A06F. To save this on wafer – it being non sequential – I first saved 0000 to 13EF and called the file TREK, then saved A04A to A06F, calling this file RANDOM. I then APPENDED TREK and RANDOM, calling this STARTREK, so when I called, the specified memory areas are loaded, leaving all other memory locations undisturbed.

All utility programs on the systems wafer are well documented, to include "default unless specified" conditions.

The obvious question is, have I had any problems? Well, my utility program "VERSION" doesn't work. This allows you to find out the version number of any utility program. The book tells you that this is a hexadecimal number stored in byte 3 of the required utility. By using the "memory change" facility of my monitor, I have examined this location of each utility, and found than all to be version 1, though why I need to know, I'm not sure. I have advised Exatron of this non-working utility and await their reply.

The utility program TTYSET appears incorrectly documented. The correct syntax, for mine at least, is TTYSET, filespec = x where x is the desired hexadecimal/decimal number.

Apart from these two tiny, and, as far as I'm concerned, totally unimportant items there have been no other problems. The system worked first, and each consecutive time, thus inspiring confidence for future use, although "ERROR MESSAGE X" has appeared as a result of fumbling on the keyboard or where I *thought* I knew the manual.

Summary

The system arrives ready to plug in and go. The quality of both boards and workmanship is far superior to that normally expected in the hobby market, and can be summed up in one word — professional. The repair service can be classed as superb, and I have no intention of trying the 30 day money back guarantee. . . I like the system too much.

The software works well and, currently, TSC BASIC, Editor and Assembler are available as optional extras; more (unspecified) are planned. Documentation is very complete, and, less the TTYSET as explained, accurate.

Exatron also produce their Stringy Floppy for S100 bus users, and both this and my own SS50 bus version derive all of the voltages necessary for use in the transport and controller from the mother board.

There's also a version available for the TRS80, though this requires mains voltage and, of course, the United States use 110 volts. Exatron do say that they can advise OEM applications not covered by the systems offered, and I'm sure it would be a very easy matter to replace the 110 volts PSU with one suitable for 240 volts. Versions are planned for both PET and Apple, though no information is available on these as yet.

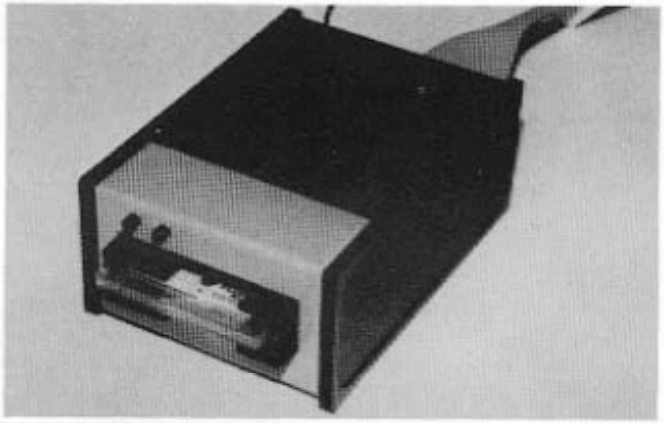
Footnote

Since I purchased my unit, ASP Microcomputers of East Malvern, Victoria, have been appointed distributors of the Stringy Floppy. Their price of $350 includes a box of ten wafers which normally retail for $3 each and a two for one extender cable to the TRS-80.

I'd recommend this system not only to every hobbyist, but also to business users as a more than economic saving over discs. Should you need almost instant access then, of course, it's not for you. But there again, who, apart from people paying for computer time by the hour, require this facility; and anyway, one of the reasons that hourly time is so expensive is the price of fast access on line storage!

It wasn't until writing this report that I looked up some of the tape wafer times I had recorded. For my first (possibly faulty) system, I judged that the system wafer took 82.5 seconds, or 7.27 inches per second which is (just) outside the -25% speed tolerance. My system now reads the wafer in 57.8 seconds, or 10.38 inches per second; maybe I could have used the troubleshooting part of the manual to better effect. In there it tells you how to either up or downgrade the transport speed, but in my innocence (ignorance) it didn't gel first off; then due to circumstances (clumsiness) beyond my control (it *was* 3 am) I didn't get a second opportunity.

Thanks to Exatron, I now have a totally redundant AC30 cassette interface with two recorders and a pile of tape software valued at over $400; this however is a small price to pay for the quality and speed I have gained. I really do think they have achieved their aim of Excellence in Electronics (mechanics too!).



MS Microsoftware

TAPE 1 LEVEL 2

Mortgage calculations, Dow Jones Industrial, cash flow, inventory-change, California income tax, journal ledger (8K), loan amortization, perpetual calendar, bio rythm, payroll, diet planning, speed reading, touch typing, sales receiptally, decision maker, mail addressing, straight depreciation, double-declining depreciation, and revolving charge account.

Also, math problems, queen, Star Trek I, number guessing, wheel of fortune, World War II bomber, rock-scissors-paper, seek, Star Trek II, Red Baron, mini-Trek, strategy, pilot, battleship, "On A Snowy Evening", Mastermind, tic-tac-toe, grand prix auto race, capitals, etch sketch, hangman.

TAPE 2 LEVEL 2

Fully documented in Some Common Basic Programs by Lon Poole & Mary Borchers:

Investment, future value regular deposits; regular withdrawals, initial minimum (for withdrawals); nominal interest, effective & earned-interest; depreciation rate, amount depreciation; salvage value; discount commercial paper; loan principal, regular and last payment, remaining balance, term-loan; mortgage amortization; greatest common denom. - integer prime factors; polygon area; triangle parts; analysis, operations two vectors; radian-degree., degree-radian conversion; co-ordinate, polar equation, functions plot; linear, curvilinear interpolation; Simpson's & trapezoidal rules, Gaussian quadrature integration; derivative.

Side 2 — quadratic equation, polynominal (Newton) & half inteval-search roots; trig, polynominal; simultaneous equations; linear programming; matrix addition, subtraction; scalar multiplication, inversion; permutations & combinations; Mann-Whitney U test; mean, variance, standard deviation; geometric mean & deviation; binomial, Poisson normal, Chi-square distribution; Chi-sq., student's T distribution test; F-distribution; linear correlation coefficient; linear, multiple-linear, Nth order, geometric, exponential regression; system reliability; future projections; Federal withholding taxes: tax depreciation schedule: check writer; recipe cost; map check; day of week; days between two dates; anglo to metric; alphabetize.

TAPE 4 LEVEL 1

Election returns, business percentage, ups and downs of business, index, inventory control, sales receipt tally, gas mileage, driving distance, mixed monthly sales report, payroll, annual earnings, speech recording aid, and double-declining depreciation.

Also, math problems, cash register, chase, snoopy, commander-in-chief, Christmas graphic, air raid, balance scale, stock market, tic-tac-toe and On A Snowy Evening.

TAPE 5 LEVEL 2

Memory test, mortgage payments, tension breaker, lineprinter-screen & vice-versa utilites, Federal income tax, election returns, business percentage, vacation planner, car pool (disk), diet planning 2, mailing list (disk) and first aid.

Also spelling bee, Star Trek 3, mind bender, tachistoscope, chase, common factor, klingon capture, spelling practice, Hamurabi, animals, Snoopy, cryptogram, starship, ants, Yesterday, and Pilot (disk). Pilot is the language of computer-aided instruction (CAI).

TAPE 7 LEVEL 2

Disassembler, Pilot, roster, dropout, memory loader, memory sort, inventory control, graph, land surveying, mixed monthly sales report, shopping list, diet planning 3, loan progress chart, hex-decimal conversion.

Also Star Trek 4, states and capitals, battleships 2, spelling practice 2, number guessing, hangman 2, snark, slot machine, cipher, target, surround, adder, termites, lunar lander, multiplication exercise, five-in-a-row, Bastem, and write. A number after a program indicates there are other similar People's Software programs. Pilot is the same as the disk pilot on tape 5, except runs on 16K tape systems.

Each Tape $9.90 (incl. postage within Australia)

Please mail your order to:

MS Microsoftware,
P.O. Box 119, Essendon, Victoria, 3040.

SELECTING YOUR FIRST MICROCOMPUTER SYSTEM

by Dr. Jon Patrick
School of Business, Prahran C.A.E.

The computer that most people first come in close contact with is quite often the small hobby microcomputer, selling for about $600 – $2000. It is most frequently sold as a home computer useful for assisting with all sorts of laborious tasks, such as maintaining your cheque account, keeping a list of recipes or for keeping track of the specials at the supermarkets. The first of these jobs is done quite efficiently by the Bank and the latter two tasks would probably be done less efficiently by the computer than by the individual. However, the quandry facing the uninformed businessman is the apparent cheapness of the "home systems" compared to the business system that is proffered to him, by a computer supplier, with a price tag of $15,000 or perhaps even $30,000.

While there are plenty of overpriced systems available on the market place there are also very real reasons for the price differences:

- The system has many more electronic components (even though outwardly the system looks similar to cheaper units).
- Individual components (both electronic and mechanical) are of a higher quality.
- Components are more complicated and sophisticated.
- System maintenance and back-up support is more comprehensive. For home computers these services may be non-existent.
- The system software is much more powerful and diverse.
- Gullible purchasers have been unable to discriminate between differently priced systems.

So, as a small business operator how do you go about deciding on the "right" system for your needs. Firstly, hire a consultant to do the assessment job for you. Secondly, if you really want to do the assessment yourself don't expect to get your system off the shelf. Rarely does a pre-existing system fit your needs precisely. However to decide on appropriate alterations to the offered package you are going to have to learn a lot about the package, then you will have to spell out carefully and in precise detail the alterations you require. Without this sort of cautious approach the system may not operate according to your requirements and so be more trouble than it is worth, and further, any alterations will be very expensive.

Don't expect to be able to make the changes yourself unless you have a great deal of computer expertise, in which case you don't need to read this article.

The businessman searching around for his first computer system is faced with two immediate problems. Firstly, what sort of price range for equipment should he be looking at, and secondly, what sort of hardware features should the system have to fulfil his needs. For any business, the basic requirements are a visual display unit with keyboard for data entry, a processor with adequate memory, mass storage medium (either floppy or hard disc) and a printer for obtaining a copy of relevant data, and most importantly the application software.

The software costs are the most difficult to ascertain and if it is to be powerful and very comprehensive it could be as expensive as the hardware. On the other hand, if you are not especially fussy and prepared to fit into someone else's design the software may only cost a few thousand dollars. Hardware costs are more tangible. For most businesses the dominant factor that affects their choice of equipment, and therefore cost, is their permanent storage requirements. Secondary factors are the maximum number of transactions per hour to be processed or the maximum rate of lines to be printed per minute.

Table 1 provides a very approximate guide to the storage requirement of five common data processing tasks. The first noticeable point is that the software demands considerable storage requirements. This fact alone removes the possibility of using the smaller (and cheaper) 5" disc drive for all but the smallest business. The minimum storage capacity for the 8" diskette is 250K characters (where K stands for 1000). It is customary and good practice to retain data files on one disc and programs on a second disc. Thus to operate the system one needs dual disc drives at a cost of about $2,500 for the pair. It should be also evident that not all the software programs can reside on one diskette. Thus changing from one task to another requires loading another diskette.

The next most important hardware item to decide on is the printer. There is an enormous variety of printers available, however a unit suitable for businesses would be a 120 characters per second matrix printer at about $2,000. If you require your output to have similar quality to say, an IBM Selectric typewriter, the printer will cost about $3,500.

The third and fourth items of hardware are the terminal, or visual display unit (VDU) and the microprocessor with associated memory. The VDU should cost between $1,000 and $1,500. The type of processor is virtually immaterial however you will need to obtain a minimum of 32K bytes of memory and preferably 48K bytes. Only if your applications programs are very large will you need to go to the maximum expansion of 64K bytes. The microprocessor and memory should cost between $1,000 and $2,000.

Stepping Up

The three sets of figures in Table 1 are intended to reflect the larger data file requirements as business operations increase in size. Those businesses with complicated requirements for which the capabilities of microcomputers are inadequate will find it necessary to use a minicomputer priced from about $25,000. A number of these complications can be clearly stated, for example:

The disc categories at the bottom of Table 1 suggest that once your stock is getting over 2000 items and you have greater than 30 staff, the flexible disc system probably will not fill your requirements. It is possible to get double-density or double-sided flexible disc drives that increase the capacity of a diskette. However for the larger business the importance of mass storage in deciding on a computer system becomes subjugated to the necessity for fast input and output of large transaction volumes. This is the case if there are more than 6 line items per invoice, or there are more than two prices per item. If your

volume of printing is great, then you may require a fast line printer costing anything from $4,000 to $12,000.

It may be necessary to input data to the computer files from a number of terminals positioned some considerable distance from the computer. In this situation communication facilities, not usually available on microcomputers, will be necessary.

Service organizations often have far greater character storage requirements than retail or production businesses. This is particularly the case if one wants to store detailed customer history records eg. electrical appliance supplier, medical practice. A single record may be as much as 500 characters. It may be necessary to search the customer file for, not only surname, but suburb or appliance. This type of data processing requires a large storage capacity, fast data retrieval and powerful search capabilities. Microcomputers tend to be inadequate in both these requirements for business environments.

With many application programs and data files spread over a large number of diskettes, staff often become frustrated with constant swapping of diskettes. This is particularly so if a file or group of programs has to be spread across two or more diskettes. Also if your data files are near the capacity of the diskette there can be problems in creating transaction files large enough for efficient processing. This overloading situation means that one has to move up to larger drives.

Conclusion

It must be re-emphasized that the software is ultimately the most important element in your complete system. It is the job of the software to mimick your manual system. If it fails to do this task to your satisfaction then you have wasted your money. On the other hand, the software can only be successful if you spend a great deal of time and trouble detailing the current system and defining the requirements of the software. The benefits gained from computerizing your operations are not instant efficiency or a sudden upturn in profitability. In a good system the benefits are a much speedier processing of trading information thereby enabling a much closer scrutiny of profitability, and a more accurate and consistent maintenance of records and accounts. So, the real benefit is substantially increased control of the information pertaining to your business. Your computer system's value to you will ultimately be measured by what you make of this control.

TABLE 1 – DATA FILE STORAGE REQUIREMENTS

Data Processing Task	Disc Program Storage Rqmt.	Chter/ Record	1 R'cord	1 Cap. K bytes	2 R'cord	2 Cap. K bytes	3 R'cord	3 Cap. K bytes
Order Entry/ Stock Inventory	140K	100	500	50	2000	200	8000	800
Creditors	100K	160	50	8	100	16	200	32
Debtors	80K	160	100	16	300	48	600	96
Payroll	60K	60	10	6	30	1.8	50	3
General Ledger	100K	60	50	5	100	10	150	15
	480K		Flexible Disc		Possibly Flex.		Hard Disc	

PASCAL. . . continued from P. 33

The coding for this procedure appears in Box 15. The entire program can now be gathered together, incorporating the extra global variables (INTERESTRATE and YEARS) into the declaration part of the first attempt (Box 10) and filling out the details of the different procedures as they have subsequently been designed.

EXERCISE:
Adapt REPAYMENTS to produce a table showing the 15 year, 20 year, 25 year and 30 year monthly repayment figures for a given range of loans. The input should be the interest rate and range of loans (and *not* the loan period) and the output should be a table with 5 columns — one for the amount of the loan and one each for each repayment period.

Conclusion

Loops control the repetition of a set of statements within a program. Every language needs a loop — PASCAL has three, which enriches the language and makes it versatile. Loops can be distinguished by the type and position of the loop test relative to the loop body.

Just as a program can be built up from basic blocks into an ordered structure, so can the data on which the program operates be organised into efficient and powerful *data structures*. The next chapter will serve as an introduction to these.

PUZZLES

LEISURE LINES

With J. J. Clessa

Many thanks to all the readers who responded to our first Leisure Lines.

Puzzle 1A involved some logical reasoning, and should not have proved much of an obstacle to our readers. The solution is that the pilot's name is SMITH.

Puzzle 1B was a bit tougher, and although it can be solved analytically, by anyone who's familiar with Diophantine analysis, it's a much simpler task to write a small program for desk calculator or microcomputer.

Using the formula for the area of a triangle (axb/2), the smallest solution is a triangle with sides 144,192,240, with a perimeter of 12^2 and an area of 24^3.

The first correct entries opened were: Puzzle 1A: Patrick Jordan of Lane Cove. Puzzle 1B: Keith De La Rue of Melbourne.

Congratulations to both and stand by for a shower of chocolate bars (not to mention the subsequent visit to the dentist).

Just one puzzle for this month, but it's really a rather interesting one.

Three friends, Alan, Bert and Colin each possess vehicles. Alan owns a big foreign car, Bert a small Australian car and Colin, a motorcycle.

One day while discussing mileages, Alan reports that his milometer, which gives 6-figure mileage readings, is currently showing a palindromic reading of 006600 miles (for those that know not, a palindromic number is one which reads the same from right to left as it does from left to right).

"What a coincidence", explains Bert, "So is mine. The 5-figure reading at the moment is 18981 miles."

"Well I never", says Colin, "although the milometer on my motorcycle only shows 4-figures, it's reading 5335 miles, which is also palindromic".

"I wonder if we're ever likely to get such a coincidence again," says Alan.

Well, of course, since each vehicle does a different weekly mileage from the others, there's no way that the question could be answered. But, supposing all three milometers were connected to just one vehicle and also supposing that they were equally accurate, then what is the least number of miles that would elapse before a) Alan's and Bert's milometers are both showing palindromic readings again? b) Alan's and Colin's milometers are both showing palindromic readings again? c) Bert's and Colin's milometers are both showing palindromic readings again? and d) all three milometers are mutually palindromic?

Answers please on a **postcard** *to Puzzle No. 3, Australian Personal Computer, PO Box 115, Carlton, 3053.*
Entries must reach our offices before August 15th.

PRIZES FOR THIS MONTH

This month's prize is really cunning. In order to make sure the winner continues to send in entries to Leisure Lines, we intend presenting him/her with a hundred 22c stamps.

Practising a little Micro-control

By Mike Dennis

Much has been said on busses and the basic schematic layout of personal computers. However, little has been published on what the various control signals that come out of the CPU chip do, and how you can use them. To list them all would fill this magazine and so I intend to fill the gap with reference to the Z80.

The basic design

Figure 1 shows the familiar block diagram of a typical computer.

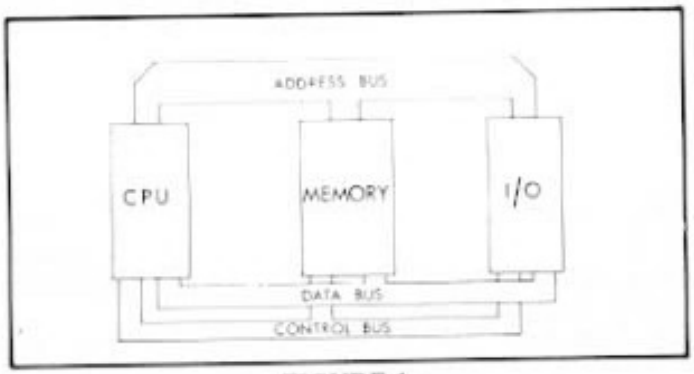


FIGURE 1.

In order to determine what control signals may be necessary, we must list those operations that are required from the computer. Table 1 shows them:

1/ CPU sends data to memory.
2/ CPU receives data from memory.
3/ CPU sends data to I/O (Input/ Output).
4/ CPU receives data from I/O.

TABLE I

Larger computers like the PDP-11 have the added facility to move data to and from memory and I/O but in the micro world this is generally not the case and so transferring data from I/O into memory would require a combination of operation (4) and (1). Table 1 can be re-arranged thus:

a/ The CPU communicates with either memory or I/O.
b/ The CPU either sends or receives data.

TABLE II

The system needs to know which one of these is to take place and the control signals do precisely that. There are four control signals from the Z80 and they are:

1/ $\overline{\text{MREQ}}$ – indicates CPU wishes to communicate with memory and that the address bus is stable and valid. ('Memory REQuest.')

2/ $\overline{\text{IORQ}}$ – indicates CPU wishes to communicate with I/O and that the address bus is stable and valid. ('I/O ReQuest.')

3/ $\overline{\text{WR}}$ – WRite – CPU wishes to send data – the data bus is now stable and valid.

4/ $\overline{\text{RD}}$ – ReaD – CPU wishes to receive data.

These signals are all automatically generated by the CPU. Moreover, as far as the CPU is concerned, they are generated at the right time with respect to each other. It is up to the user to do something useful with them!

For example, when operation (2) is in progress, the CPU would make both $\overline{\text{MREQ}}$ and $\overline{\text{RD}}$ active. The bar over the top means that they are 'active low', i.e. when they are active, they are at logic 0 or 0v or whatever you like to call it. Table III shows the logic state of the four control signals for each of the operations.

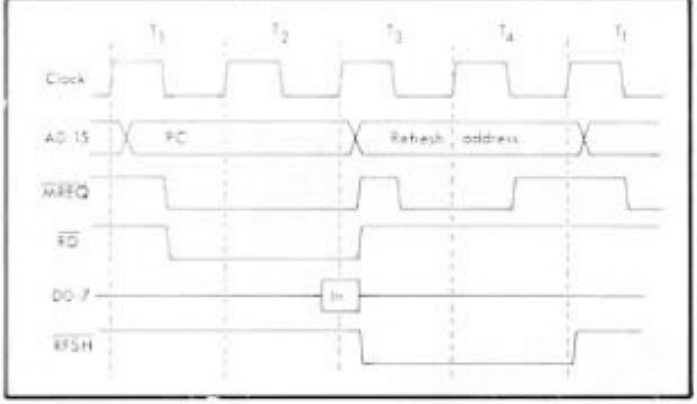


FIGURE 2.

(Table I operations)

	MREQ	IORQ	RD	WR
(1)	0	1	1	0
(2)	0	1	0	1
(3)	1	0	1	0
(4)	1	0	0	1

TABLE III

Some computer manufacturers, e.g. Tandy, gate these controls together and derive four different ones:– $\overline{\text{MEMRD}}$ and $\overline{\text{MEMWR}}$ (memory read and write) and $\overline{\text{IORD}}$ and $\overline{\text{IOWR}}$. Unfortunately, this results in the loss of flexibility of the four original signals.

Use of control signals

The basic concept of the tri-state bus is that there is only one device on the bus at any one time that is supplying any data – the talker. All the other devices are sitting across the bus and listening. The control signals help discriminate between the talker and the listeners and allow them to talk or listen at the correct time.

After reset is hit, all the internal registers are cleared and the program counter is forced to a specific value or address, depending on which CPU chip is used. The contents of the program counter are then transferred to the address bus as an address and, in the case of the Z80, this address is 0000. The Z80, therefore, expects to find the first program instruction at this address. As this instruction will reside in memory and the CPU wants to read it, so $\overline{\text{MREQ}}$ and $\overline{\text{RD}}$ will be made active (in this case, active low) by the CPU – see Figure 2, which has been taken from the Z80 Technical manual. Notice that the whole operation takes four clock cycles.

The next step is to apply the controls to memory in such a way that the data at this address 0000 finds its way onto the data bus so that the CPU can read it.

How memory uses control signals

Figure 3 shows the pin connections for a 2708 EPROM. This device has a capacity of 1K bytes (2^{10} bytes) and so 10 pins are needed to access all the memory locations within the chip.

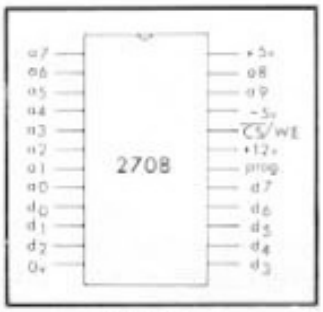


FIGURE 3.

The data is output onto the data bus via pins 9-11 and 13-17 inclusive only when pin 20 (CS; CS = Chip Select) has been taken low otherwise the o/ps (outputs) of the 2708 are tri-stated. The CPU has put out 0000_H or 0000 0000 0000 0000 onto the address bus but the 2708 is only interested in the logic state of the ten lower bits of the bus (A_0 to A_9). The remaining six bits can be used to uniquely define the 2708's position in the possible 64K bytes of memory that the micro could address. There are two methods of decoding:

1/ Full decoding

Here the remaining six bits are combined via a logic circuit that will only give an output when a certain bit pattern is present. The 2708 must respond to a base address of 0000 and so the circuit of Figure 4 will decide this. The output from the decoder is 0 which, conveniently, is needed by CS in order to enable the 2708.

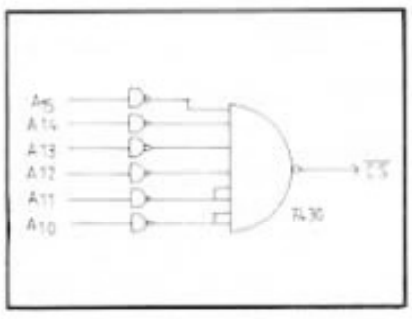


FIGURE 4.

2/ Partial decoding

This is only suitable for a system of limited size and one that is not going to expand. Assuming that it consists of, say, 4K bytes of RAM and 2K bytes of ROM, then each of the spare six address lines could be used directly as the Chip Select (CS) for each K bytes of memory. Figure 5 shows how they could all be used and the various base addresses that they will each respond to. Notice that address 0000 still needs to be completely decoded and this has not been shown. Care must be taken when programming to prevent any attempt at reading or writing to addresses that would select two or more blocks of memory. For example, an instruction to read data at address F000 would be disastrous as all four blocks (A, B, C & D) would be selected simultaneously!

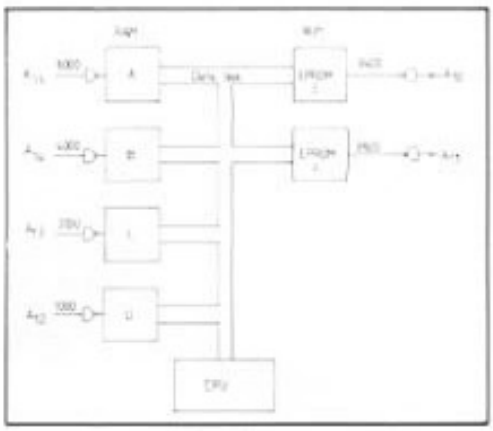


FIGURE 5.

Full decoding is certainly well worthwhile and always pays dividends in the end as bus conflicts are avoided. You perhaps are wondering about $\overline{\text{RD}}$ and $\overline{\text{WR}}$? Well . . . .

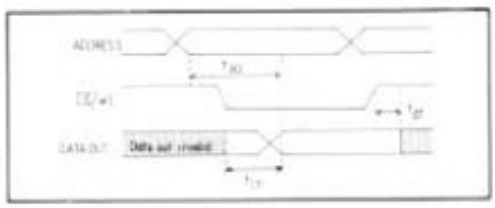


FIGURE 6.

All memory has an inherent delay between the address being presented at the memory chip's pins and then valid data appearing on the output pins. There will almost certainly be rubbish on the pins until a certain time has passed – the access time. Access time is often quoted as part of the chip number, eg – 2102-4 (450nS) access time. It can also be found from data sheets, and Figure 6 shows that for the 2708, while Table IV gives the timings.

Symbol	Parameter	Min.	Typical	Max	Unit
t_{acc}	Address to o/p valid		280	450	nS
t_{co}	Chip select to o/p valid		60	120	nS
t_{df}	Chip select to o/p tri-stated			120	nS

after the address has been presented and CS has been low for at least 120 nSecs. Relate Figure 6 to Figure 2. If we are using a clock frequency of 2MHz then one clock cycle will take 0.5uS or 500nSecs. The CPU samples data on the rising edge of T3 and so our data must be valid and stable on the data bus by this time.

Have we enough time? The address is present on the 2708 for nearly all of T2 and most of T1 before the critical edge of T3 occurs. That is nearly 1000 nSecs and more than enough for our 2708. We can use Figure 4 to chip select the 2708 and further combine this signal with $\overline{MREQ}$ and $\overline{RD}$ to enable a separate chip that buffers the outputs of the 2708 from the data bus. This is shown in Figure 7.

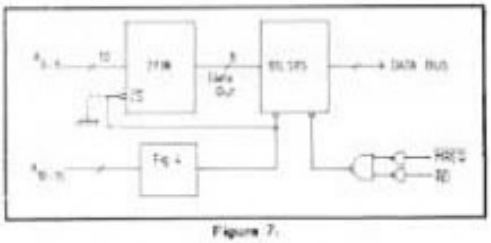


Figure 7.

In fact, with Figure 7, there is no reason why CS of the 2708 shouldn't be permanently tied low provided that the tri-state buffer is only enabled when $A_{10} - A_{15}$ are low and $\overline{MREQ}$ and $\overline{RD}$ are active – shown dotted. A similar process can be applied to RAM and this is discussed next.

Read and write

RAM really is an awful word! We don't actually random access the memory; if we did, there's no telling what data would come out. What is done is Read/Write but no-one seems to call it that. Figure 8 shows the timing diagram for a Memory Write by the Z80; Figure 9 – the timing diagram for a 2102 RAM; and table V gives the timings.

Symbol	Parameter	Min. Time (nSec)
t_{wc}	Write cycle	450
t_{aw}	Address to write set-up	20
t_{wp}	Write pulse width	300
t_{dw}	Data setup	300
t_{cw}	Chip enable (select) to write setup	300
t_{dh}	Data hold time	

TABLE V

A brief explanation of some of the above follows now:

t_{aw} – the address must be valid for at least 20nSec before write goes low.

t_{dw} – the data must be valid 300 nSec before the write pulse goes high.

t_{dh} – data must remain valid for 0nSec after write goes high.

signals from the Z80 are capable of providing the times shown in Table V. There is a fair margin as well. For example, data remains valid for some time (almost 200nSecs) after write goes high and so there is plenty of room for a chip that has a longer t_{dh}. Figure 10 shows how a 2102 could be connected up to a buffer and address decoder. Work out what base address it will respond to – answer given at the end*.

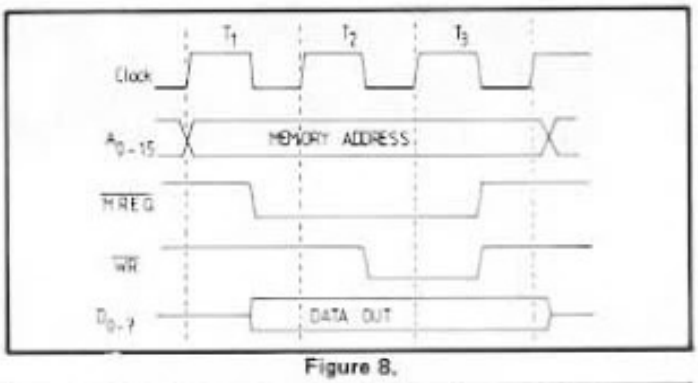


Figure 8.

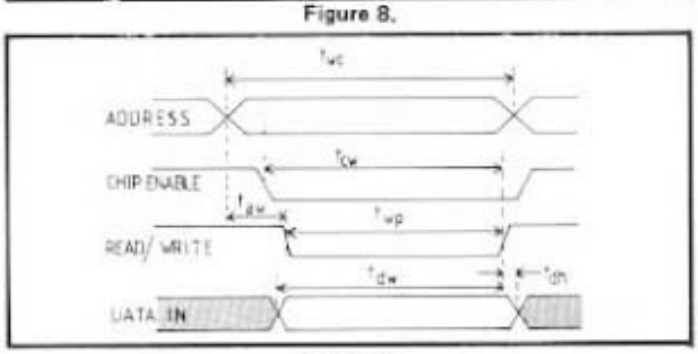


Figure 9.

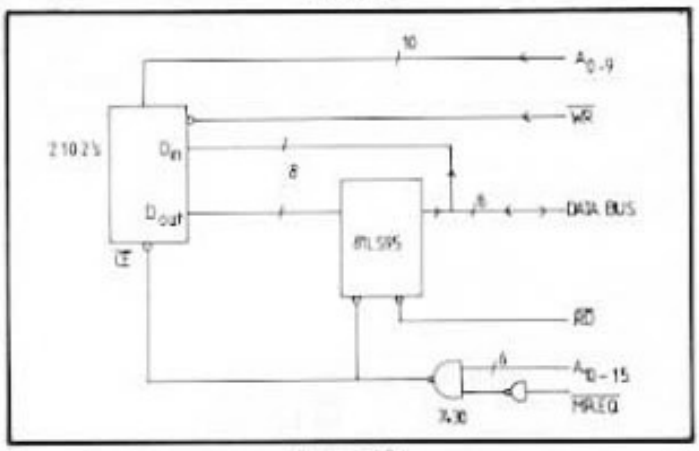


Figure 10.

What about I/O?

So far, we have only considered memory. I/O is very similar but this time, as mentioned previously, $\overline{IORQ}$ is used and not $\overline{MREQ}$. $\overline{RD}$ and $\overline{WR}$ still perform the same functions. However, only the bottom eight address lines are used to decode which port is being accessed. Eight lines allow 256 different ports to be decoded and $\overline{RD}$ and $\overline{WR}$ will tell them which direction the data is going. Sometimes, several ports exist within the one chip (e.g. PIO's which have two data ports and at least two command or status ports) and the chip performs the decoding for itself for devices that transfer data slowly, like UARTS, are easy, whereas those that transfer data quickly, like CRT or floppy disc controllers, are more difficult. You have to get the data sheets, burn the midnight oil and try it for yourself.

That completes this quick look at control signals but one further point remains regarding convention. When the CPU chip is reading data from either memory or I/O, then that device is writing to the CPU, so should the $\overline{RD}$ or $\overline{WR}$ line go active? Convention states that it is the action of the CPU that dictates which line goes active and, in this case, it will be the $\overline{RD}$ line – common sense really, since it's the CPU that generates the control signals and it should know what it's doing!

*(Ans: FCOO)

PROGRAMMING - THE SIMPLE APPROACH

Mervyn Axson leads you gently through the minefield of writing your first "real" programs in BASIC. Suitable for businessmen as well as hobbyists, you need only a nodding acquaintance with the language and access to a machine in order to start.

The operations to be performed in most business programs are very simple, so the programming should also be simple. You may doubt the truth of this if you look at a listing of a program, for at first sight it probably appears to be very complicated. If, however, you examine it bit by bit, you will find that it really is quite simple. I'll be tackling the problem in the reverse way, by writing a very basic program, and then refining it step by step to show how it ends up looking complicated. And by the way, although the program will be written in PET's version of BASIC, it can easily be modified to suit other machines.

A problem common to many businesses is that of quoting credit terms to prospective customers. Let us suppose that we require a minimum deposit of 10% and that the credit charges are at the rate of 12½% per annum calculated on the balance remaining after payment of the deposit. The query is: "What are the terms for good costing $399.95 over 18 months?". The calculations are not difficult, although perhaps tedious.

Now we have to consider how to write the computer program to carry out these operations for us. Actually, the simple answer is that we have already done so, for these are all valid BASIC statements! We only have to add line numbers and allow for inputting Sum, Interest Rate and Period, and outputting Deposit and Monthly repayment. Our completed program reads:

```
10 INPUT S
20 INPUT I
30 INPUT P
40 D = S/10
50 B = S - D
60 C = B * (I/100 * (P/12))
70 A = B + C
80 R = A/P
90 PRINT D
100 PRINT R
```

Note that this short and simple program is complete in itself and will give the required answers for any values that you wish to input. You could stop here, since most of the refinements to be added are largely cosmetic in that they do not improve on the basic function, the output, or make the program easier to use. Of course, in business, these factors can be very important.

What happens if we load the program into PET and then type RUN? PET responds with a "?" so type in the sum (followed by pressing the RETURN key, of course). Another "?" appears, and you type in the interest rate, and finally in response to a third "?" you type in the period. PET now displays the answers, and the screen looks like this:

```
RUN
? 399.95
? 12.5
? 18
  39.995
  23.7470313
```

We would of course mentally round the 39.995 to 40.00 and 23.7470313 to 23.75, but we can easily make PET do it for us. INT(X*100+.5)/100 will round X to 2 decimal places. In this program we only need to use this twice, but in others we have to use it many times so to save typing we can use the DEF statement:

```
5 DEF FNA(X) = INT(X*100+.5)/100
```

and then

```
90 PRINT FNA(D)
100 PRINT FNA(R)
```

will produce the required result.

It would be helpful if PET told us what input it wanted rather than just printing "?". INPUT statements allow us to do this. If we amend line 10 to:

```
10 INPUT "COST OF GOODS" ; S
```

PET will now print:

```
COST OF GOODS?
```

Lines 20 and 30 can also be modified similarly, and lines 90 and 100 can be altered to:

```
90 PRINT "DEPOSIT";FNA(D)
100 PRINT P; "MONTHLY PAYMENTS
    OF";FNA(R)
```

The output will now be:

```
DEPOSIT 40
18 MONTHLY PAYMENTS OF 23.75
```

This is beginning to look better, but there is still a lot that can be done to improve it. Showing the Deposit as 40 rather than 40.00 is annoying to say the least. BASIC does this to all numbers, dropping zeros that are not significant. Typing PRINT 40.10 will produce 40.1 and PRINT 00123 results in 123. There is a fairly simple way round this problem, although it looks complicated! We convert our simple number to a string which can then be "formatted" to produce the desired result and then printed. We will probably need to use this many times in a lot of business programs, so we can write the program section as a subroutine which we can call up whenever required.

This is:

```
10000 Z$ = STR$(Z)
10010 L = LEN(Z$)-2
10020 IF L = 0 THEN 10060
10030 IF MID$(Z$,L,1) =
      "." THEN 10090
10040 L = L+1
10050 IF MID$(Z$,L,1) =
      "." THEN 10080
10060 Z$ = Z$+".00"
```

Deposit = $399.95 divided by 10 = $40.00
Balance = $399.95 minus $40.00 = $359.95
Charges = Balance multiplied by interest rate multiplied by period.
Interest rate = 12½% p.a. the period = 18 months.
The rate to be applied = 12½%* 1.5 = 18¾%
So charges = $359.95 multiplied by 18.75/100 = $67.49
Amount to be repaid = balance plus charges = $359.95 plus $67.49 = $427.44.
Divide this last figure by 18 to find the monthly payment which is $23.75.

So for the general case we can say:	or, just using initials:
Deposit = Sum/10	D = S/10
Balance = Sum – Deposit	B = S – D
Charges = Balance * (Interest * Period/12)	C=B* (I/100* (P/12))
Amount to be repaid = Balance + Charges	A = B + C
Repayments = Amount to be repaid/Period.	R = A/P

10090 RETURN

The number to be printed is returned as Z$, but before we amend the program, there is one further point in formatting. The result would be clearer if the amounts were set out thus:

```
DEPOSIT                          40.00
18 MONTHLY PAYMENTS OF           23.75
```

Whilst in this case a simple TAB(30) instruction would be satisfactory, it would not if the deposit was 10.00 and payments 9.95:

```
DEPOSIT                          10.00
18 MONTHLY PAYMENTS OF            9.95
```

There is a very simple way to align the numbers. They are in string form so LEN(Z$) works out how long they are and TAB (30 − LEN (ZS)) will ensure that they are lined up.

Now the program reads:

```
88 Z = FNA(D)
89 GOSUB 10000
90 PRINT "DEPOSIT";
   TAB(30—LEN(Z$));Z$
98 Z = FNA(R)
99 GOSUB 10000
100 PRINT P;"MONTHLY PAYMENTS
    OF";TAB(30—LEN(Z$));Z$
```

We now have a fairly presentable result on PET's VDU. A few PRINT statements judiciously inserted would make it even better, since for example, a simple 91 PRINT will put a line space between DEPOSIT and 18 MONTHLY PAYMENTS, making the result easier to read. We want this program to provide clear answers for the novice, so what else should we do? For a start, it would help them to know that they had loaded the correct program into PET. 4 PRINT "LOAN SCHEME" will reassure them. Then we could clear the VDU screen when we have input the data so that the answers alone are shown. PRINT CHR$(147) will clear the screen and as we have now lost the input data perhaps we should also display the cost of the goods in the output.

```
81 PRINT CHR$(147)
82 PRINT "LOAN SCHEME"
83 PRINT
84 Z = FNA(S)
85 GOSUB 10000
86 PRINT "COST OF GOODS";
   TAB(30 − LEN (Z$)); Z$
87 PRINT
```

Just for one moment, let us suppose that we are now satisfied with our efforts and that we think the program to be complete. We "RUN" it with the sample data mentioned before and PET immediately displays:

```
LOAN SCHEME
COST OF GOODS                   399.95
DEPOSIT                          40.00
18 MONTHLY PAYMENTS OF           23.75
```

We now write down the figures and return to the customer and proudly announce the results. Our efficiency must be obvious and surely we will make the sale? Alas, real life is not like that and we are soon deflated for the response is: "Oh! I can pay $100.00 down, so what would that make the monthly payment?" The program that we have written does not allow for this of experience, we could modify the program. Instead of using line 40 to calculate the deposit, we could make an input of the deposit offered:

```
40 INPUT "DEPOSIT OFFERED";D
```

However, think for a minute; sometimes the response to the question: "What deposit do you want to pay?" will not be a definite figure but "What is the least that you require?" We can cater for both responses simply by adding:

```
31 INPUT"DEPOSIT OFFERED ?
   IF LOWEST TYPE MIN";D$
32 IF D$ = "MIN" THEN 40
```

We have made the variable D$ rather than D to cater for the input of MIN. If it is, then line 32 continues the program as before. But if D$ represents an amount e.g. 100.00, then we must convert this from a string variable D$ to a numeric variable D. D = VAL(D$) will do this and we can then go straight to line 50.

It could happen that the deposit offered was less than the minimum required and this may not be noticed; we can add a line to take care of this as well.

```
34 IFD<S/10THENPRINT"MINIMUM DEPOSIT IS";FNA(S/10):GOTO31
35 GOTO 50
```

If you try running the program now, you will find that whilst it produces the required result, it also ends by printing RETURN WITHOUT GOSUB ERROR. This is because after line 100, which is the end of the program so far, PET "falls through" to the subroutine in 10000 on. A simple line 9999 END will prevent this. We could usefully clear the screen before any input is requested, so line 3 PRINT CHR$(147) can be added.

The listed program does now begin to look rather more complicated, but by taking it a step at a time, it's been written quite painlessly. And what's more, we have a program which has been capable of being used at any stage in its development — which is by no means completed yet. How about adding the facility of being able to output the payments required for all of the periods we offer, which could typically be 12, 18, 24, 30 and 36 months? We will tackle this and other developments in the next section. In the meantime, we have a quite useful program already. In a business situation it is often import-

First listing of program 'LOAN SCHEME'

```
3 PRINTCHR$(147)
4 PRINT"LOAN SCHEME"
5 DEFFNA(X)=INT(X*100+.5)/100
10 INPUT"COST OF GOODS";S
20 INPUT"INTEREST RATE";I
30 INPUT"PERIOD";P
31 INPUT"DEPOSIT OFFERED ? IF LOWEST TYPE MIN";D$
32 IFD$="MIN"THEN40
33 D=VAL(D$)
34 IFD<S/10THENPRINT"MINIMUM DEPOSIT IS";FNA(S/10):GOTO31
35 GOTO50
40 D=S/10
50 B=S-D
60 C=B*(I/100*(P/12))
70 A=B+C
80 R=A/P
81 PRINTCHR$(147)
82 PRINT" LOAN SCHEME"
83 PRINT
84 Z=FNA(S)
85 GOSUB10000
86 PRINT" COST OF GOODS";TAB(30-LEN(Z$));Z$
87 PRINT
88 Z=FNA(D)
89 GOSUB10000
90 PRINT" DEPOSIT";TAB(30-LEN(Z$));Z$
91 PRINT
98 Z=FNA(R)
99 GOSUB10000
100 PRINTP ;"MONTHLY PAYMENTS OF";TAB(30-LEN(Z$));Z$
9999 END
10000 Z$=STR$(Z)
10010 L=LEN(Z$)-2
10020 IFL=0THEN10060
10030 IFMID$(Z$,L,1)="."THEN10090
10040 L=L+1
10050 IFMID$(Z$,L,1)="."THEN10080
10060 Z$=Z$+".00"
10070 GOTO10090
10080 Z$=Z$+"0"
10090 RETURN
```

People's Pascal

$29·95

At last your TRS-80* can run Pascal too! The Chung/Yuen "tiny" Pascal is fully implemented for Level II TRS-80*, 16K and up. You no longer need to be left out of the growing group of Pascal users, because People's Pascal gives you everything you need to write structured Pascal programs:

• tiny Pascal compiler • complete text editor for writing your programs • complete tiny Pascal monitor • sample Pascal programs • users manual (TRS-80 Computing issue 1:4)

People's Pascal is both a powerful, structured language and "CPU expeditor". People's Pascal programs execute at least four times faster than Basic, and often eight-times faster! Special functions open up the complete graphic capability of TRS-80*. You now have the means to write those dazzling, impressive, high-speed graphics programs that are great for games, plotting, statistics, etc.

For the serious computerist, side two of People's Pascal II (tape 6) contains a larger compiler and complete source to the compiler, written in Pascal. This means you can re-compile the compiler, making changes, adding features, etc. (but this will take at least 36K RAM and a solid knowledge of programming).

With the complete People's Pascal operating system, you can save and load both source (Pascal) programs, and compiled programs, to or from cassette tape. This means that once you have de-bugged a program, you can save the P-code (compiled program) and thereafter, to run the program, you need only load the super-fast P-code.

Here is a partial list of People's Pascal features: recursive procedure/functions • for (loop) • case if/then/else • one-dimensional arrays • write • read constant • repeat/until(loop) • "peek & poke" • plot (graphics for TRS-80*).

MS MICROSOFTWARE

Please send mail order to P.O. Box 119, Essendon, Victoria, 3040.

15 Beryl Street, West Essendon.

the system in which they've invested.

The next stage in developing our loan scheme program is to add the option of calculating the payments required for all the periods we offer, which we will take to be 12, 18, 24, 30 and 36 months. This will enable us to answer the general query: "What are your terms for .....?" quickly and comprehensively. This development does need a bit of thinking about. The input is easy, being very similar to that used for the deposit option. We shall have to alter some line numbers for reasons that will appear later, so delete line 30 and write:

```
25 INPUT "PERIOD ?
   IF ALL TYPE ALL"; P$
26 IF P$<>"ALL" THEN P = VAL(P$)
```

Now we either have a single value for the period in P or we have "ALL" in P$. If the former, then the program as it stands will work; but what changes are needed to cope with the latter? What we have to do is to run through lines 60, 70 and 80 for each value of the period. This obviously calls for a "FOR ..... NEXT" loop. BASIC does allow us to specify the step between values, as well as the start and finish, so FOR P = 12 to 36, STEP 6 will successively give the correct values to P. But what happens in line 80? Each time we run through the loop a new value will be calculated for R which will replace the previous one, so when we exit from the loop the only value of R available is the last one i.e. R for 36 months. Obviously, we must arrange to save the value of R each time it is calculated. Fortunately, BASIC provides an easy way to do this, although you may not think so from the jargon, for we construct an array using a single subscripted variable (or something like that!) All this actually means is that we save the first result as variable R(1), the second as R(2) and so on. The program changes the number in brackets, the subscript, each time we go round the loop, the result being that our five values for R are all saved as R(1) to R(5).

How do we write the program to perform these operations? We will use J to keep track of the subscript and first we give it the value of 1, 55 J = 1, then we set up the loop, 57 FOR P = 12 TO 36 STEP 6. Lines 60 and 70 remain the same, but we alter 80 to R(J) = A/P and give it the new line number of 77. R(1) now has the value of R when P = 12. J now has to equal 2 so 79 J = J+1. We now have to go back to the beginning of the loop, line 57, to do the calculation for P = 18. 80 NEXT will accomplish this, and since J now equals 2, the result will be saved as R(2). This will be repeated until all five results have been saved and we exit from the loop to line 81.

We have now dealt with the case when P$ = "ALL", but what when P has a single value? This would work previously, but now we have altered the program by putting in a loop — which in this instance we don't need! We must miss out the loop instructions and a couple of IF . . . . . THEN statements in lines 56 and 78 will be sufficient. The complete section of the

```
56 IF P$<>"ALL" THEN 60
57 FOR P = 12 to 36 STEP 6
60 C = B * (I/100(P/12))
70 A = B + C
77 R(J) = A/P
78 IF P$<>"ALL" THEN 81
79 J = J + 1
80 NEXT
```

This may seem a little complicated at first, but once you get the idea it is really quite simple. It's well worth making the effort to fully understand it, since it is a technique that is very valuable in many business programs, where the ability to perform repeated calculations and later recall the results is a necessity.

We have just mentioned recalling the results, so how do we do that to produce our output. Very simply, just by using the same technique.

```
95 J = 1
96 IF P$<>"ALL" THEN 98
97 FOR P = 12 TO 36 STEP 6
98 Z = FNA(R(J))
99 GOSUB 10000
100 PRINT P; "MONTHLY PAYMENTS
    OF"; TAB(30 − LEN(Z$));Z$
104 NEXT
```

You may remember that we left a few spare lines after inputting the period. This is to allow us to check that the data input is valid. As the program stands, it will perform the calculations for any period of months typed in, even 1 or 1000. Admittedly, these are unlikely errors, but 21 instead of 12 or 42 instead of 24 are very possible. There are many ways in which we carry out this check, but a simple one can be based on the fact that all the valid periods can be divided by 6, resulting in whole numbers (integers) ranging from 2 to 6. We can code this in BASIC in two lines. IF P/6 ◊ INT(P/6) THEN "ERROR" checks for whole numbers e.g. 21/6 = 3.5 an error. 42/6 = 7 will pass this test but will fail IF P/6 < 2 OR P/6 > 6. Actually, as often happens, the program becomes a little simpler if we reverse the last test to IF P/6 = (6AND P/6) = 2 THEN "PROCEED AS NORMAL". An error e.g. 42/6 = 7 will then carry on to the next statement which is the error message, so saving another jump statement.

```
Sample runs of program "LOAN SCHEME"

LOAN SCHEME QUOTATION BY C.AXSON & SONS

 COST OF GOODS                          399.95

 DEPOSIT                                 40.00

 18 MONTHLY PAYMENTS OF                  23.75        INCLUDING CHARGES OF 67.49


LOAN SCHEME QUOTATION BY C.AXSON & SONS

 COST OF GOODS                          659.00

 DEPOSIT                                159.00

 12 MONTHLY PAYMENTS OF                  46.88        INCLUDING CHARGES OF 62.50

 18 MONTHLY PAYMENTS OF                  32.99        INCLUDING CHARGES OF 93.75

 24 MONTHLY PAYMENTS OF                  26.04        INCLUDING CHARGES OF 125.00

 30 MONTHLY PAYMENTS OF                  21.88        INCLUDING CHARGES OF 156.25

 36 MONTHLY PAYMENTS OF                  19.10        INCLUDING CHARGES OF 187.50
```

Loan scheme — final version

```
3 PRINTCHR$(147)
4 PRINT"LOAN SCHEME"
5 DEFFNA(X)=INT(X*100+.5)/100
10 INPUT"COST OF GOODS";S
20 INPUT"INTEREST RATE";I
25 INPUT"PERIOD ? IF ALL TYPE ALL";P$
26 IFP$<>"ALL"THENP=VAL(P$):GOTO 28
27 GOTO31
28 IFP/6<>INT(P/6)THEN30
29 IFP/6=<6ANDP/6>=2THEN31
30 PRINT"INVALID PERIOD":GOTO25
31 INPUT"DEPOSIT OFFERED ? IF LOWEST TYPE MIN";D$
32 IFD$="MIN"THEN40
33 D=VAL(D$)
34 IFD<S/10THENPRINT"MINIMUM DEPOSIT IS";FNA(S/10):GOTO31
35 GOTO50
40 D=S/10
50 B=S-D
55 J=1
56 IFP$<>"ALL"THEN60
57 FORP=12TO36STEP6
60 C(J)=B*(I/100*(P/12))
```

```
29 IF P/6 = (6 AND P/6)=2 THEN 31
30 PRINT "INVALID PERIOD" : GOTO 25
```

There is one further valuable addition we could make to the program, and that is to give the option of having the output printed out. Devices external to PET are given numbers and that for a printer is usually 4. To output to the external device, a file is opened — OPEN 1,4 is the code. Now any statement starting PRINT #1, will cause the output to be sent to that device. So after enquiring whether the option is wanted and receiving the answer "yes" we run through a series of statements identical to lines 82 to 104 but with PRINT #1, instead of PRINT. Actually, they are not quite identical for you will notice that the TAB instructions are slightly different. This is caused by the way the printer used, a Teletype 43, responds to the TAB instruction. On PET, TAB(30) causes printing to start in the 30th print position from the beginning of the line, but on the Teletype TAB(30) causes printing to start in the 30th print position from where the print head is situated. This means that if we have already printed DEPOSIT, printing will start in print position 37 and not 30 as required. We therefore have to deduct the length of any items already printed, including spaces between words, from the required position number e.g. TAB(23) after DEPOSIT.

Another addition has been made to the printed output. If the quotation is given to the customer, we should show the credit charges which would be made in each case for the differing periods. This information is present in variable C and we can use the same technique to preserve the values as we did for R, i.e. C(J) is substituted for C in lines 60 and 70. For clarity we want to print the credit charges on the same line as the repayments, but we cannot give the instructions in the same statement line since we have to GOSUB to format C(J). By ending the payment print instruction with a ";" we suppress the carriage return and line feed, so achieving our object. The only other alteration to the program occurs in line 102 where we alter THEN 9999 to THEN 110 to give the printed output option.

We now have a fairly comprehensive answer to our problem, which even the most inexperienced junior can use to give quick and accurate answers to queries. The full listing does show quite a complicated piece of programming, and I certainly would not like to have to sit down write it all at one go. However, I hope you have seen that it really is not all that complicated if broken down into steps, as I have done. The experts may scorn my methods, but they meet the criteria I have laid down. The program works and it does just what I want. It operates quickly enough for the output to be shown both on the VDU and the printer at the fastest they will operate. No doubt it could be made more elegant, but time is short and there are, no doubt, many other tasks to perform.

```
77 R(J)=A/P
78 IFP$<>"ALL"THEN81
79 J=J+1
80 NEXT
81 PRINTCHR$(147)
82 PRINT" LOAN SCHEME"
83 PRINT
84 Z=FNA(S)
85 GOSUB10000
86 PRINT" COST OF GOODS";TAB(30-LEN(Z$));Z$
87 PRINT
88 Z=FNA(D)
89 GOSUB10000
90 PRINT" DEPOSIT";TAB(30-LEN(Z$));Z$
91 PRINT
95 J=1
96 IFP$<>"ALL"THEN98
97 FORP=12TO36STEP6
98 Z=FNA(R(J))
99 GOSUB10000
100 PRINTP ;"MONTHLY PAYMENTS OF";TAB(30-LEN(Z$));Z$
101 PRINT
102 IFP$<>"ALL"THEN110
103 J=J+1
104 NEXT
110 PRINT"DO YOU REQUIRE PRINTED RESULT ?"
120 PRINT"IF YOUR ANSWER IS YES, THEN SWITCH"
130 PRINT"THE TELETYPE ON AND LOAD PAPER"
140 PRINT"PRESS DATA BUTTON ON TELETYPE AND"
150 PRINT"TYPE 1 ON PET KEYBOARD"
160 PRINT"OTHERWISE TYPE 2"
170 INPUTH
180 IFH<>1THEN9999
190 OPEN1,4
200 PRINT#1,"LOAN SCHEME QUOTATION BY C.AXSON & SONS"
210 PRINT#1
220 Z=FNA(S)
230 GOSUB10000
240 PRINT#1," COST OF GOODS";TAB(37-LEN(Z$));Z$
250 PRINT#1
260 Z=FNA(D)
270 GOSUB10000
280 PRINT#1," DEPOSIT";TAB(43-LEN(Z$));Z$
290 PRINT#1
300 J=1
310 IFP$<>"ALL"THEN 330
320 FORP=12TO36STEP6
330 Z=FNA(R(J))
340 GOSUB10000
350 PRINT#1,P ;"MONTHLY PAYMENTS OF";TAB(28-LEN(Z$));Z$;
360 Z=FNA(C(J))
370 GOSUB10000
380 PRINT#1,TAB(10);"INCLUDING CHARGES OF";Z$
390 PRINT#1
400 IFP$<>"ALL"THEN9999
410 J=J+1
420 NEXT
9999 END
10000 Z$=STR$(Z)
10010 L=LEN(Z$)-2
10020 IFL=0THEN10060
10030 IFMID$(Z$,L,1)="."THEN10090
10040 L=L+1
10050 IFMID$(Z$,L,1)="."THEN10080
10060 Z$=Z$+".00"
10070 GOTO10090
10080 Z$=Z$+"0"
10090 RETURN
```

VICTORIA

AUSOM

Apple Users' Society of Melbourne. Contact Mr David Turk of Computerland Melbourne.

Commodore Computer Users Association

The newly inaugurated Commodore Computer Users Association of Victoria will meet regularly at 7.30 p.m. on the last Tuesday of the month at the North Melbourne Football Club Social Club, Fogarty Street, North Melbourne. The Association has a variety of aims in order to support users of Commodore Microcomputers and to use such media as newsletters, seminars, conferences and the like to inform members of the latest developments. For further details, please contact Nicki Saunders, The Secretary, on 614-1433 or 614-1551 during business hours.

Compucolor Users' Group

Write to Mr. L. Ferguson of 12 Morphett Avenue, Ascot Vale for all the information necessary.

Geelong Computer Club

Interested people should contact Mr Peter McKeon, P.O. Box 93, Geelong, 3220

S.C.U.A.

Sorcerer Computer Users (Australia). Further details may be obtained from the Secretary, S.C.U.A., P.O. Box 144, Doncaster, 3108.

S.M.U.G.

To find out more about this group of SORD M100 users, contact Mr Robin Miller, 60 Winmalee Drive, Glen Waverley, 3150.

NEW SOUTH WALES

Commodore User Group

The Commodore User Group of Sydney considers itself "an effective method of sharing current information, ideas, programming techniques, hardware interfacing, and cost effective applications relating to the Commodore computers between commercial users and hobbyists and the manufacturer". A monthly newsletter is available to members, with product news, details of current software and a User's Directory to other sources of information.
For more details, drop a line to Mr John Guidice, C/- The Commodore Users Group, G.P.O. Box 4721, Sydney, 2001.

Compucolor Users' Group

If you are interested, Andrew MacIntosh of 91 Regent Street, Chippendale, is the man to see.

80AT

80 Applications Transfer. A new group formed to serve all users of 80 series microprocessors (8080, 8085, Z80).
The group will function as a software exchange and applications forum for Australian 80 series microprocessor users from all applications spheres, including industrial, scientific, business and hobbyist. A newsletter will be published bi-monthly and get togethers will be held periodically.
80AT has software from the CP/M Group available for distribution and is currently negotiating to gain access to the PROTEUS software library and other applications libraries. Software currently available from the 80AT software resources ranges from simple I/O programs through system utilities up to complete languages such as STOIC, ALGOL-M and FELIX. As 80AT is strictly a spare time activity for the convenors it would be appreciated if all enquiries are by mail only: To 80AT, C/- Planet 3 Systems, 47 Birch St., Bankstown, NSW 2200.

QUEENSLAND

Brisbane Youth Computer Group

Mr. A. Harrison, P.O. Box 396, Sunnybank, 4109, should be contacted for more information.

IREE Microcomputer Interest Group

Enquiries should be directed to Mr. N. Wilson, P.O. Box 81, Albion, 4010.

SOUTH AUSTRALIA

TRS80 Users' Group

To obtain details contact Mr. G. Stevenson of 36 Sturt Street, Adelaide, 5000

A.C.T.

MICSIG

Further information concerning MICSIG, from the Registrar, MICSIG, C/- P.O. Box 446, Canberra City, 2601.

NEW ZEALAND

Wellington Microcomputer Society Inc.

Write to Lindsay Williams, 2 Pope Street, Plimmerton, New Zealand.

FAX

THE Z80 MNEMONICS ARRANGED BY OP CODE Compiled by John A Coll.

MSB LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NOP	DJNZ d *	JRNZ, d *	JR NC, d *	LD B,B	LD D,B	LD H,B	LD (HL), B	ADD B	SUB B	AND B	OR B	RET NZ	RET NC	RET PO	RET P	0
1	LD BC, nn	LD DE, nn	LD HL, nn	LD SP, nn	LD B,C	LD D,C	LD H,C	LD (HL), C	ADD C	SUB C	AND C	OR C	POP BC	POP DE	POP HL	POP AF	1
2	LD (BC), A	LD (DE), A	LD (nn), HL	LD A, nn	LD B,D	LD D,D	LD H,D	LD (HL), D	ADD D	SUB D	AND D	OR D	JP NZ, nn	JP NC, nn	JP PO, nn	JP P, nn	2
3	INC BC	INC DE	INC HL	INC SP	LD B,E	LD D,E	LD H,E	LD (HL), E	ADD E	SUB E	AND E	OR E	JP nn	OUT A,port	EX (SP), HL	DI	3
4	INC B	INC D	INC H	INC (HL)	LD B,H	LD D,H	LD H,H	LD (HL), H	ADD H	SUB H	AND H	OR H	CALL NZ, nn	CALL NC, nn	CALL PO, nn	CALL P, nn	4
5	DEC B	DEC D	DEC H	DEC (HL)	LD B,L	LD D,C	LD H,L	LD (HL), L	ADD L	SUB L	AND L	OR L	PUSH BC	PUSH DE	PUSH HL	PUSH AF	5
6	LD B,n	LD D,n	LD H,n	LD (HL), n	LD B, (HL)	LD D, (HL)	LD H, (HL)	HALT	ADD (HL)	SUB (HL)	AND (HL)	OR (HL)	ADD n	SUB n	AND n	OR n	6
7	RLCA	RLA	DAA	SCF	LD B,A	LD D,A	LD H,A	LD (HL), A	ADD A	SUB A	AND A	OR A	RST 0	RST 10H	RST 20H	RST 30H	7
8	EX AF, AF *	JR d *	JR Z,d *	JR C,d *	LD C,B	LD E,B	LD L,B	LD A,B	ADC B	SBC B	XOR B	CP B	RET Z	RET C	RET PE	RET M	8
9	ADD HL, BC	ADD HL, DE	ADD HL, HL	ADD HL, SP	LD C,C	LD E,C	LD L,C	LD A,C	ADC C	SBC C	XOR C	CP C	RET	EXX	JP (HL) *	LD SP, HL	9
A	LD A, (BC)	LD A, (DE)	LD HL, (nn)	LD A, (nn)	LD C,D	LD E,D	LD L,D	LD A,D	ADC D	SBC D	XOR D	CP D	JP Z,nn	JP C,nn	JP PE, nn	JP M,nn	A
B	DEC BC	DEC DE	DEC HL	DEC SP	LD C,E	LD E,E	LD L,E	LD A,E	ADC E	SBC E	XOR E	CP E	See Manual *	IN A,port	EX DE, HL	EI	B
C	INC C	INC E	INC L	INC A	LD C,H	LD E,H	LD L,H	LD A,H	ADC H *	SBC H	XOR H	CP H	CALL Z,nn	CALL C,nn	CALL PE, nn	CALL M,nn	C
D	DEC C	DEC E	DEC L	DEC A	LD C,L	LD E,L	LD L,L	LD A,L	ADC L	SBC L	XOR L	CP L	CALL nn	See Manual *	See Manual *	See Manual *	D
E	LD C,n	LD E,n	LD L,n	LD A,n	LD C, (HL)	LD E, (HL)	LD L, (HL)	LD A, (HL)	ADC (HL)	SBC (HL)	XOR (HL)	CP (HL)	ADC n	SBC n	XOR n	CP n	E
F	RRCA	RRA	CPL	CCF	LD C,A	LD E,A	LD L,A	LD A,A	ADC A	SBC A	XOR A	CP A	RST 8	RST 18H	RST 28H	RST 38H	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Note
* means code not implemented on 8080
Flags are not affected by load, 16 bit increment and 16 bit decrement instructions
n means a single byte
nn means a double byte
d means a relative displacement in 2s complement form

Op codes CB, DD, ED and FD include all bit operations
JUMP and CALL mnemonics are conditioned by: NZ — non zero; NC — non carry, PO — parity odd; P — sign positive; Z — zero; C — carry; PE — parity even; M — sign negative (minus)
(HL) means that the contents of the address given in HL etc . . .

AUSTRALIA'S MOST UP-TO-DATE BUYER'S GUIDE FOR MICROCOMPUTERS

Month by month, every effort will be made to keep
In Store up-to-date and accurate.
And that means APC will always be happy to hear from its readers
of any errors, and additions that seem worthy of inclusion.

LIST OF ABBREVIATIONS

A – Assembler
A/D – Analog to Digital
B/W – Black and White
C – Cassette
cps – Characters per second
Doc – Documentation
E – Extensive
Ed – Editor
Ex – Extended
F/D – Floppy Disc
H – Hardware
I – Introductory
I/O – Input / Output
int – Interface
K/B – Keyboard
M/A – Macroassembler
N/A – Not Available
N/P – Numeric Pad
O/S – Operating System
P/P – Parallel Port
RAM – Random Access Memory
ROM – Read Only Memory
res – Resolution
S – Software
S/P – Serial Port
T/E – Text Editor
U – Utility
VDU – Video Display Unit

All prices shown are exclusive of sales tax, except where indicated by an asterix.

Software items listed in *italics* are not included in the basic price of the equipment.

Name of Machine	Main Distributor & Phone No	Hardware	Software	Doc	Price	Comments
Apple II plus	Computerland (03) 62 5581 (02) 29 3753	16-48K RAM: 6502: colour VDU int.: 81/O slots: games paddles: option 5¼" F/D (116K) and 11 MB disc	O/S: BASIC: *Pascal:* games	E	$1395	280x192 high res colour graphics: Applesoft BASIC in 12K ROM
Century	Abacus Computer Store (03) 429 5844	C100, 48-64K RAM: Z80: 12" VDU: 2x5¼" F/D (2x143K): 112 cps printer: RS232 port: S100 bus: C200 includes 2xF/D (2x315K): hard disc: 4xRS232: 2xP/P	*COBOL: FORTRAN: BASIC*	I	C100 – $4950: C200 – $5400	Also available: C300
Challenger IP	Systems Automation (02) 439 6477	4-32K RAM: 6502: C int: 24x32 VDU int: RS232 port: option – dual 5¼" F/D (140K)	O/S: BASIC: A: games	I	$448	8K microsoft BASIC in ROM: expansion board available
Challenger 4	Systems Automation (02) 439 6477	8-48K RAM: 6502: colour 32x64 VDU int: RS232 port: P/P: option – 6502C microprocessor; dual 5¼" F/D (140K)	BASIC: *Pascal*	I	$871	BASIC in 8K ROM
Compucolor II	Anderson Digital Equipment (03) 543 2077	8-32K RAM: 8086: 13", 32x64 8 colour VDU: single 5¼" F/D (51K): RS232 port	ExBASIC(ROM):*A*	I	$2095	16K model, $2395: 32K $2695: maintenance manual available
Cromemco System 2, System Z2H, System 3	–	64-512K RAM: Z80A: System 2, dual 5¼" F/D (346K): System Z2H, also Winchester disc (11MB): System 3, 8" dual 1MB: S/P: P/P	CDOS: *BASIC: COBOL: FORTRAN: M/A: ExBASIC: Structured BASIC*	E	System2 $3990 System Z2H $9650 System 3, $6750	All Systems expandable to multi user (2-7 users) $2880–$8825
Exidy Sorcerer Model II	Dick Smith Electronics (02)888 3200	8-48K RAM: Z80: 30x64 VDU int.: RS232 Port: P/P: S100 bus:extra C int.	O/S: ExBASIC (ROM): *M/DOS: CP/M*	I	$1295*	High res graphics capability: 16K version $1395*: 32K version $1525*: 48K version $1655*: User programmable character set

Name of Machine	Main Distributor & Phone No	Hardware	Software	Doc	Price	Comments
HP-85	Hewlett Packard Australia (03)89 6351	16-32K RAM: N/A: 5", 16x32 B/W VDU: C(200K): 64 cps printer: RS232 port: 4 x P/P	BASIC	S	$3550	Full dot matrix graphics: N/P: compact portable unit
IPS-100	Microprocessor Applications (03) 754 5108	32-896K RAM: 8085: 2 RS232 ports: S100 bus: dual 5¼" F/D (630K)	O/S: Ex BASIC: Ed: A: *CP/M: CBASIC: FORTRAN: COBOL*	E	$3750	
Microengine	Daneva Control (03)598 9207 Abacus Computer Store for complete system (03) 429 5844	64K RAM: MCP 1600: 2 x RS232 ports: 2xP/P: Options – dual 5¼" F/D (Single or dble density); 8" F/D (single or dble density)	BASIC: Pascal: File Manager: U	E	$2995	Also available as board
North Star Horizon	Melbourne's Byte Shop (03) 568 4022	32-64K RAM: Z80A: 5¼" F/D (170K): 2xS/P: 1P/P: optional – VDU ($1350); Quad density F/D	DOS: BASIC: *COBOL: FORTRAN: Pascal: CP/M: M/A*	E	$2695	
Pet 2001	Hanimex (02) 938 0400	8-32K RAM: 6502: C: 9" 25x40 B/W VDU: extra C Int: IEEE488 port	O/S: BASIC: *A*	I	8K$1199 16K$1859 32K$2249	Options – dual 5¼" F/D (353K), $2329: $109 for disc operating ROM
Sord M100 ACE III	Alliance Digital Corporation (02) 436 1600 Abacus (03) 429 5844	48K RAM: Z80: 24x64, 12" VDU: RS232 ports: 2x5¼" F/D (2x143K): S100 bus: 2 octave speaker: A/D Conv.: option–8 colour graphic controller ($1450)	O/S: Ex*BASIC FORTRAN*	I	$4500	M100 ACE IV – 8 colour graphics controller incl.
Sord M223	Alliance Digital Corporation (02) 436 1600 Abacus (03) 429 5844	64K RAM: Z80: 12", 24x80 VDU: 2xRS232 port: S100 bus: 5¼" F/D (350K)	O/S: *ExBASIC: FORTRAN: COBOL*	I	$7500	
TRS-80 Level 1	Tandy Electronics (02) 638 6633	4-16K RAM: Z80: C: 12", 16x64 B/W VDU	BASIC: Games: A	I	$699*	BASIC in 4K ROM: upgradable to Level 2
TRS-80 Level 2	Tandy Electronics (02) 638 6633	4-48K RAM: Z80: C:12", 16x64 B/W VDU: RS232 port: P/P	BASIC: *M/A: FORTRAN: COBOL*		$879*	16K machine includes N/P: 4-16K upgrade $320*: ($250* without N/P): max. config. $1169*: option – single 5¼" F/D (78K), (max of 4)
Vector Graphics System B	AJ & JW Dicker (02) 524 5639	64K RAM: Z80: Dual 5¼" F/D (630K): 12", 24x80 B/W VDU: S/P: 2xP/P	DOS: BASIC: A: CP/M: Ed	E	$6350	Graphics and numeric pad.
Versatile 4	Microprocessor Applications (03) 754 5108	32-56K RAM: 8085: 9", 24x80 B/W VDU: dual 5¼" F/D (630K): S100 bus: 2xRS232	MBASIC: MDOS (including T/E and A): *Version 4 MDOS AND BASIC: CP/M*	E	$5692	

SINGLE BOARDS

Acorn	Cottage Computers (03) 481 1975	1-8K RAM: 6502: EPROM socket: Hex K/B: C int: 8 digit LED display: up to 16 ports: options – Eurocard 64 way connector; VDU card; Full K/B card	½K monitor: *BASIC*	S&H	System1 $285: SystemII $1224: SystemIII (incl.a5¼" F/D), $2763	Universal interface card available.
Aim 65	Dwell Pty Ltd (02) 487 3111	1-4K RAM: 6502: 8K ROM: full K/B: 20 character LED display: 20 character thermal printer: Cx2 int: 1 P/P	8K monitor in ROM: *A: BASIC*	E	$525*	Case available $75*
SBC100	Microtrix (03) 718 2581	1K RAM: Z80: 8K ROM: S100 bus: 1S/P: 1P/P	1K monitor: DOS in ROM	E	$299	Also available assembled $374
Superboard	Systems Automation (02) 439 6477	4-32K RAM: 6502: 10K ROM: full K/B: 24x32 VDU Int: C int: options–RS232; dual 5¼" F/D (140K)	BASIC: games	I	$360	BASIC in 8K ROM

Byte Saver

by Peter Dillon

Written for a Level II TRS-80, *Byte Saver* is a good adjunct to the many BASIC renumbering programs which use spaces to compensate for shorter line numbers being inserted in the program text (eg. 1030 might be replaced by 430). It is also a good way of tidying-up and compressing programs during their development.

Rather than requiring memory size to be set, *Byte Saver* POKEs its machine code into the string in line 65300. The memory position of the routine is located using the VARPTR function and the reserved word, NAME, is used to call the routine. Whenever the interpreter encounters the word NAME, it jumps to a machine language routine who's entry point is given by the values at memory locations 16783 and 16784.

To use *Byte Saver*, first CLOAD the short merge program and RUN it; then CLOAD the program requiring compression. After it has loaded type POKE 16548, 233: POKE 16549, 66 and RUN again. Now the *Byte Saver* can be loaded and RUN.

Byte Saver will remove all spaces in the program text except those following REM statements or between inverted commas.

MERGE

```
0 K=PEEK(17129)+PEEK(17130)*256
1 J=K:K=PEEK(J)+PEEK(J+1)*256:IFK=0THENPOKE16548,J-INT(J/256)*256:POKE16549,INT(
J/256)ENDELSE1
```

BYTE SAVER

```
65000 CLS:NEXLIN=17129:P=PEEK(16548)+PEEK(16549)*256
65005 PRINTCHR$(23):PRINT@128,"ORIGINAL MEM REQ.";P-17129:PRINT:PRINT"NO. BYTES
TO BE CHECKED"
65010 GOSUB65300
65020 IFNEXLIN=NOMOVES=PTHENPOKENEXLIN,0:POKENEXLIN+1,0:POKE16548,233:POKE16549,
66:PRINT:PRINT"MEMORY SAVED":P-NEXLIN:END ELSE GOSUB65100
65040 MP=MP+1:IFPEEK(MP)=0ORPEEK(MP)=147THEN65020
65050 IFPEEK(MP)=34THENFLAG=FLAG+-1
65060 IFPEEK(MP)=32ANDFLAG=-1THENGOSUB65200
65070 PRINT@304,P-MP-NOMOVE-2:GOTO65040
65100 REM     **** FINDS MEMORY POSITION OF NEXT LINE ****
65110 MP=NEXLIN:OLDLIN=NEXLIN
65120 NEXLIN=PEEK(MP)+PEEK(MP+1)*256
65130 MP=MP+3:FLAG=-1:RETURN
65200 REM     **** MOVES BLOCK OF MEMORY ****
65205 REM          AND ADJUSTS POINTERS
65210 NUMBYTES=P-MP-1:SOURCE=MP+1:DESTIN=MP:NOMOVE=NOMOVE+1:MP=MP-1:NEXLIN=NEXLI
N-1
65240 POKEK+2,SOURCE-INT(SOURCE/256)*256:POKEK+3,INT(SOURCE/256)
65250 POKEK+5,DESTIN-INT(DESTIN/256)*256:POKEK+6,INT(DESTIN/256)
65260 POKEK+8,NUMBYTES-INT(NUMBYTES/256)*256:POKEK+9,INT(NUMBYTES/256)
65270 NAME:POKEK+2,1:POKEK+5,1:POKEK+8,1:POKEK+9,1:HD=OLDLIN
65280 HA=PEEK(HD)+PEEK(HD+1)*256-1:POKEHD,HA-INT(HA/256)*256:POKEHD+1,INT(HA/256
):IFPEEK(HA+NOMOVE+2)+PEEK(HA+NOMOVE+3)*256=65000THEN RETURN ELSEHD=HA:GOTO65280
65300 Z$="12345678901234"    **** LOADS M/L ROUTINE ****
65310 DATA217,33,2,2,17,1,60,1,2,2,237,176,217,201
65320 J=VARPTR(Z$):K=PEEK(J+1)+PEEK(J+2)*256
65330 FORL=KTOK+13:READM:POKEL,M:NEXT
65340 POKE16783,PEEK(J+1):POKE16784,PEEK(J+2):RETURN
```


SOFTWARE

AUSTRALIA'S MOST COMPREHENSIVE CATALOGUE

SEAHORSE

supplies more than 470 programmes from:

MUSE / PROGRAMMA / MICROSOFT / MICROSENSE / LIFEBOAT / GEMSOFT

for the

APPLE

CBM 'PET'

TRS-80

SORCERER

$1.00 inc. p & p refundable with order

Please specify your computer.

SEAHORSE COMPUTER SERVICES

P.O. Box 47, Camden, N.S.W. 2570

Old S. Rd. Razorback, N.S.W.

Telephone: (046) 266 121

TRS-80 Graphics

by Peter Dillon

The following program uses among other routines, a machine language subroutine to provide for very rapid movement of large number of bytes.

In its wisdom, Microsoft did not include a MOVE command and in doing so forfeited an opportunity of creating a language capable of rapid manipulation of large blocks of memory. The routine used in *TRS-80 Graphics* is fourteen bytes long and includes the size of the memory block to be moved, source address and destination address. In this case the block of memory is set at 1024 bytes corresponding to a full video screen. Source and destination addresses are POKEd into the routine according to whether information is to be transferred to the screen from memory or vice versa. Lines 1000 to 1030 POKE the machine code into memory beginning at 21980 and memory size should be set to this value. Memory from 22000 to 32767 is used to store ten screens numbered 0 through 9. Rather than tying up the USR command, *Graphics* utilizes the reserved word NAME which is assigned an entry point in line 1020. The USR command is therefore free to be used for other machine language routines.

The *Graphics* program is designed for a 16K Level II machine and uses the numeric keypad's two left columns (i.e. keys 1, 2, 4, 5, 7 and 8) to correspond to the six pixels per video byte. Key "Ø" will fill the current cursor position and "." will clear it. The four arrows are used to move the flashing cursor in their respective directions.

To move a screen to memory, press key "M" and then the desired memory number (0-9). The contents of any memory can be called upon by pressing the "R" key and then the appropriate number (0-9). Once created, a screen can be saved on tape. First put the screen in a memory (by typing for eg. "M4") then press the "S" key followed by "4". Ensure the cassette player is in the record position before executing the memory to cassette routine. Should you press the "S" or "I" key accidently you will have lost the current screen but can recover the command mode by pressing the key "E". The recorded screen can be input from the tape by typing "I" followed by the memory to be used to store the screen.

The Draw subroutine can be accessed by typing "D". It provides an easy way of developing extensive layouts on screen by compensating for another of Microsoft's omissions, the PLOT function. In addition to this, circles may be produced by indicating centre and radius. No more than 15 of each of circles and lines may be requested at any one program run and each line or circle is specified by a number from 0 to 14.

A summary of executable routines is given by typing shift "H" while in the command mode.

```
5 REM          ))) GRAPHICS PROGRAM (((
10 CLS:CLEAR500:DEFSTR:DIM A(17):DIM L(14,4):DIM E(14,3):P=15360:SA=21982:DA=21
985
12 DATA 55,56,52,53,49,50,48,46,91,10,8,9,10,77,82,83,73,68
14 FORX=0TO17:READA:A(X)=A:NEXT
16 GOSUB1010
18 REM COMMAND MODE
20 I=INKEY$:IFI=""THENE=PEEK(P):POKEP,191:POKEP,E:GOTO20 ELSE GOSUB25:GOTO20
25 A=ASC(I)
30 FORX=0TO17:IFA(X)<>ATHENNEXT:RETURN
35 ONX+1GOSUB40,60,40,40,40,40,40,40,40,82,84,86,88,70,105,105,2000,3010,4000
38 RETURN
39 REM MOVES SCREEN INDICATOR
40 IFPEEK(P)<32THENPOKEP,PEEK(P+2+X):RETURN ELSEPOKEP,128+2+X:RETURN
42 POKEP,191:RETURN
44 POKEP,32:RETURN
62 IFP<15424THENP=P+64
63 RETURN
64 IFP<16320THENP=P+64
65 RETURN
66 IFP<15744THENP=P-1
67 RETURN
68 IFP<16384THENP=P+1
69 RETURN
70 CLS:PRINTCHR$(23)
72 PRINT:PRINT"'M' MOVE SCREEN TO MEMORY":PRINT"'R' RECALL SCREEN FROM MEMORY":P
RINT"'S' SAVE SCREEN ON TAPE":PRINT"'I' INPUT SCREEN FROM TAPE":PRINT"'D' GOTO D
RAW ROUTINE":PRINT"SHIFT 'H' FOR HELP ROUTINE":PRINT:PRINT"PRESS ANY KEY TO CONT
INUE"
74 GOSUB10000:CLS:RETURN
100 REM MOVES SCREEN TO MEMORY AND VICE VERSA
105 IFI="M"THENMI=1ELSEMI=0
106 GOSUB10000
110 IFASC(I)<48ORASC(I)>57RETURN
140 MA=22000+VAL(I)*1024:ME=MA-INT(MA/256)*256:MC=INT(MA/256)
145 IFMI=1THENPOKESA,0:POKESA+1,60:POKEDA,ME:POKEDA+1,MC:NAME ELSEPOKESA,ME:POKE
SA+1,MC:POKEDA,0:POKEDA+1,60:NAME
240 P=15360:IFMI=1THENCLS
250 RETURN
1000 REM LOADS M/L ROUTINE
1010 DATA 217,33,0,60,17,1,60,1,0,4,237,176,217,201
1020 POKE16763,220:POKE16764,85
1030 FORX=21980TO21993:READY:POKEX,Y:NEXT:RETURN
1999 REM MEMORY TO TAPE ROUTINE
2000 CLS:PRINT@960,"PRESS ANY KEY TO CONTINUE"
2010 GOSUB10000:IFI="E"THENCLS:RETURNELSECLS
2020 PRINT"WHICH SCREEN DO YOU WANT SAVED":GOSUB10000:X=VAL(I)*1024+22000:CLS
2030 FORZ=0TO3:A$="B":FORY=0TO239:A$=A$+CHR$(PEEK(X+Y+Z*240)):NEXTY:A$=A$+"E":PR
INTMID$(A$,2,240):PRINT#-1,A$:NEXTZ
2040 A$="B":FORY=0TO63:A$=A$+CHR$(PEEK(X+960+Y)):NEXTY:A$=A$+"E":PRINTMID$(A$,2,
64):PRINT#-1,A$:RETURN
3000 REM TAPE TO MEMORY ROUTINE
3010 CLS:PRINT"PRESS ANY KEY TO CONTINUE":GOSUB10000:IFI="E"THENCLS:RETURNELSECL
S
3020 PRINT"INTO WHICH MEMORY DO YOU WANT THE SCREEN TO BE PUT":GOSUB10000:X=VAL(
I)*1024+22000:CLS
3030 FORY=0TO3:INPUT#-1,A$:A$=MID$(A$,2,240):PRINTA$;:FORZ=0TO239:POKEX+Y*240+Z,
ASC(MID$(A$,Z+1,1)):NEXTZ,Y
3040 INPUT#-1,A$:A$=MID$(A$,2,64):PRINTA$;:FORY=0TO63:POKEX+960+Y,ASC(MID$(A$,Y+
1,1)):NEXT:RETURN
4000 REM MENU FOR DRAWING, SAVING etc., SCREENS
4010 CLS:PRINTCHR$(23):PRINT:PRINT"TYPE:":PRINT"'C' TO ORDER CIRCLES":PRINT"'L'
TO ORDER LINES":PRINT"'D' TO DRAW LINES/CIRCLES":PRINT"'E' TO ESCAPE TO COMMAND
MODE ":PRINT"'A' TO SEE A LIST OF ALL ORDERS":GOSUB10000
4020 IFI="L"THENGOSUB4040ELSEIFI="C"THENGOSUB4100ELSEIFI="D"THENGOSUB4500:RETURN
ELSEIFI="A"THENGOSUB4555ELSEIFI="E"THEN CLS:RETURN
4025 GOTO4010
4040 CLS:INPUT"PLEASE ENTER LINE NUMBER ";X:L(X,0)=1
4050 INPUT"PLEASE ENTER LINE NUMBER'S ORIGINATING COORDINATES ";Y1,Y2
4060 INPUT"AND TERMINATING COORDINATES ";Y3,Y4
4070 L(X,1)=Y1:L(X,2)=Y2:L(X,3)=Y3:L(X,4)=Y4
4080 RETURN
4100 CLS
4110 INPUT"PLEASE ENTER CIRCLE NUMBER";N
4120 INPUT"PLEASE ENTER DESIRED RADIUS";R
4130 INPUT"PLEASE ENTER COORDINATES OF CENTER OF CIRCLE";X,Y
4140 C(N,0)=1:C(N,1)=R:C(N,2)=X:C(N,3)=Y:RETURN
4500 CLS
4510 FORX=0TO14:IFL(X,0)<>0THENNEXT:GOTO4522
4520 DY=L(X,4)-L(X,2):DX=L(X,3)-L(X,1):IFDX=0THENFORY=L(X,4)TOL(X,2)STEPSGN(L(X,
2)-L(X,4)):SET(L(X,1),Y):NEXTY,X:GOTO4522
4522 J=DY/DX:K=L(X,2)-J*L(X,1)
4524 FORX1=0TO127
4526 IFX1<L(X,1)ORX1>L(X,3)THENNEXTX1,X:GOTO4572
4528 Y=J*X1+K:IFY<48ANDY>=0THENSET(X1,Y):NEXTX1,X:GOTO4572
4530 NEXTX1,X:GOTO4522
4532 FORM=0TO14:IFC(M,0)<>0THENNEXT:RETURN
4534 FORY=C(M,2)-C(M,1)TOC(M,2)+C(M,1)
4536 FORN=-1TO1STEP2
4538 XSQUARED=C(M,1)^2-(Y-C(M,2))*(Y-C(M,2))*2:IFXSQUARED<0THENNEXTN,Y,M:RETURNELSE X=(SQ
R(XSQUARED))+2.18*N+C(M,2):IFX<0ANDX<127THENSET(X,Y):NEXTN,Y,M:RETURN ELSE NEX
TN,Y,M:RETURN
4555 CLS:PRINT"LINE NO.","ORIG. COORDS.","FINAL COORDS."
4560 FORX=0TO14:IFL(X,0)=0THENNEXTELSEPRINTX,L(X,1);L(X,2),L(X,3);L(X,4):NEXT
4570 PRINT@960,"PRESS ANY KEY TO CONTINUE";:GOSUB10000
4580 CLS:PRINT"CIRCLE NO.","CENTER COORDS.","RADIUS"
4590 FORX=0TO14:IFC(X,0)=0THENNEXTELSEPRINTX,C(X,2);C(X,3),C(X,1)/2:NEXT
4600 PRINT@960,"PRESS ANY KEY TO CONTINUE";:GOSUB10000:CLS:RETURN
10000 I=INKEY$:IFI=""THEN10000ELSERETURN
```

DATASOFT
MICROCOMPUTER SOFTWARE

Proudly present the largest range of software in the world for the **COMMODORE MICRO COMPUTER.**

"PETSOFT"
Over 200 programs Including:- Over 40 Business Programs Over 65 Programming Aids & Tutorial Programs. Over 60 Simulations and Games Programs.

- **Both cassette and disc based programs available.**
- **No hassle guarantee - if our programs don't load, we will replace free of charge.**
- **Write or phone now for our fabulous free catalogue.**

DATASOFT
139 Guildford Road,
MAYLANDS W.A. 6051.
Tel: **(09) 271 7169**
Agents required all States.

Word Processor

The following listing is suitable for an 8K Exidy Sorcerer.
Our thanks to Tony Hailes.

```
10 CLEAR4500
11 PRINTTAB(10);"SIMPLE WORD PROCESSER"
12 PRINT
15 DIMA$(100)
16 FORA=1TO5:READA1:NEXT
17 FORA=0TO14
19 READA1
20 POKEA,A1
21 NEXTA
25 DATA76,80,65,70,67
27 DATA245,253,126,61,246,128,253,119,69,211,254,241,195,18,224
45 PRINT"TYPE IN YOUR TEXT, EACH LINE STARTING WITH A ";CHR$(34)
;"."
46 PRINT"AT THE END OF EACH LINE, PRESS RETURN."
47 PRINT"WHEN YOU HAVE FINISHED, TYPE [END] ."
50 N=N+1
60 INPUTA$(N)
80 IF A$(N)<>"[END]"THEN50
85 N=N-1
90 PRINT:PRINT:PRINT
100 PRINT"YOUR LETTER....."
110 PRINT
115 PRINT"LINE"
120 FORA=1TON
130 PRINTA;CHR$(34);A$(A)
140 NEXTA
150 PRINT
160 PRINT"_____
_____"
161 PRINT
165 IFB=1THEN240
170 PRINT:PRINT"EDITING."
180 PRINT"COMMANDS:"
190 PRINT:PRINT"LIST - TO LIST LETTER"
200 PRINT"PRINT - TO PRINT LETTER ON PRINTER"
210 PRINT"<LINE NUMBER> - TO CHANGE THAT LINE"
211 PRINT"-<LINE NUMBER> - TO DELETE THAT LINE"
212 PRINT"<LINE NUMBER>+0.5 - TO INSERT A LINE"
213 PRINT"ADD - TO ADD LINES ONTO THE END"
214 PRINT"FIND - FINDS ANY SEARCH STRING IN THE TEXT"
215 PRINT"CHANGE - TO INSERT OR DELETE ANYTHING IN ONE LINE"
220 PRINT
230 B=1
240 INPUT"COMMAND";A$
250 A=VAL(A$)
255 IFA<>0THEN290
256 RESTORE
260 FORA1=1TO5
270 READA2
275 IFASC(A$)=A2THEN279
277 NEXTA1
278 GOTO180
279 ONA1GOTO100,500,50,750,820
280 IFA=0THEN240
290 IFA>NTHENN=A
292 IFA<0THEN690
295 IF INT(A)<>ATHEN630
300 PRINTA-1;CHR$(34);A$(A-1)
310 INPUT" ";A$(A)
330 PRINTA+1;CHR$(34);A$(A+1)
340 GOTO240
500 POKE8142,0
510 A1=PEEK(8144)
520 A2=PEEK(8145)
530 POKE8144,0
540 POKE8145,0
550 NULL4
555 PRINT:PRINT:PRINT:PRINT
560 FORA=1TON
570 PRINTA$(A)
580 NEXTA
590 POKE8144,A1
600 POKE8145,A2
610 NULL0
620 END
630 FOR I=NTO(INT(A)+1) STEP-1
640 A$(I+1)=A$(I)
650 NEXT
660 N=N+1
670 A=INT(A)+1
680 GOTO300
690 A=ABS(A)
700 FOR I=A+1TON
710 A$(I-1)=A$(I)
```

```
720 NEXT
730 N=N-1
740 GOTO240
750 INPUT"ENTER SEARCH STRING: ";B$
760 B=LEN(B$)
770 FORI=1TON
780 GOSUB1000
782 IFC$=""THEN800
784 PRINTI;C$;PRINTI;
785 IFT=1THEN795
786 FORJ=1TOT-1
788 PRINT" ";
790 NEXTJ
795 PRINT"↑"
800 NEXTI
810 GOTO240
820 INPUT"ENTER LINE NUMBER";I
830 INPUT"ENTER OLD TEXT";B$
835 B=LEN(B$)
840 GOSUB1000
850 IFC$<>""THEN865
860 PRINTI;A$(I)
870 PRINT"ERROR - TEXT NOT FOUND"
880 GOTO240
885 INPUT"ENTER NEW TEXT";C$
887 C=LEN(C$)
888 A$=A$(I)
890 A$(I)=MID$(A$,1,T-1)+C$
892 IFT-1+B>ATHEN900
895 A$(I)=A$(I)+MID$(A$,T+B,A-B-T+1)
900 PRINTI;A$(I)
910 GOTO240
1000 C$=""
1010 A=LEN(A$(I))
1020 FORT=1TOA-B+1
1030 IFMID$(A$(I),T,B)<>B$THEN1060
1040 C$=A$(I)
1050 GOTO1070
1060 NEXTT
1070 RETURN
READY
```

Space Slalom

by Geoffey Salt
and Steve Withers

This program, for Apple II, is written in Applesoft, and makes use of the low resolution colour graphics facility.

The object of the game is to get a spacecraft from the bottom of the screen to the space station at the top, without being hit by any of the asteroids, and without hitting the sides of the space station. You move the spacecraft using paddle 1. The program interprets its value in one of three ways: with the control fully anticlockwise the craft will step left; with it fully clockwise the carft will step right; with it central it will not move horizontally at all. In addition, if the pushbutton on Paddle 1 is pressed the craft will move up.

The program allows up to five craft to reach the space station, but this can be altered if desired by changing line 470. At each iteration the outline of the craft is checked to see if it has changed colour; if so, then it has been hit by an asteroid, or has collided with the space station. If the craft is hit it is destroyed and a new one must be launched. If there are none left then you are doomed. Moreover, each craft has only just enough fuel to reach the space station. The fuel level is shown by a red bar at the left of the screen. If this bar falls to zero then the craft will stop, and will be helpless until hit by an asteroid. (If it stops in the space station, but without docking, then the game is lost, as it cannot be removed in time).

The area in front of the space station is protected so that no asteroids appear there. Similarly, asteriods do not appear on top of the spacecraft at the start of the game.

```
10  GOTO 400
20  REM HIT CHECK
30  IF F4 = C0 AND PD = 19 THEN F20: REM DOCKED SUCCESSFULLY
40  X = SCRN( PD+F3, F4 ) : SCRN( PD+F4, F4 ) : SCRN( PD+F4 + C1, F4 ) : SCRN( PD + C1+F4, F4 ) : SCRN( PD + C1+F4,
50  X = X + SCRN( PD, C1+F4 + C1 ) : SCRN( PD + C2+F4, F4 ) : SCRN( PD + C2+F4 + C1, F4 ) : SCRN( PD + C2, F3 )
60  IF X = 0 THEN RETURN
70  GOSUB 850
80  XX = XX - C1: IF XX < C0 THEN 1220
90  TEXT : HOME : PRINT "THE SPACECRAFT HAS BEEN DESTROYED.": PRINT
100 PRINT "YOU ONLY HAVE "; C1; " SPACECRAFT LEFT."
110 PRINT : PRINT "PRESS ANY KEY TO LAUNCH THE NEXT ONE"
120 POP : GOTO 480
130 REM CRAFT FLIP
140 F = - F5: C0
150 IF FU < 47 AND - ABS( UT - FU ) < C1 AND F4 = C3 THEN 1270: REM BLOCKING ENTRANCE
160 IF FU < 47 THEN RETURN : REM NO FUEL
170 VT = C0: HK = C0: REM MOVE DIRECTION
180 IF PEEK ( - 16287) > 127 AND F4 < C0 THEN VT = - C1
190 IF F = "S" AND FU < C1 THEN HK = - C1
200 IF F = " " AND FU < 47 THEN HK = C1
210 IF VT = C0 AND HK = C0 THEN RETURN : REM NO MOVE
220 COLOR= C0: GOSUB 2500: PLOT C0+FU*PD + FU + C1, F4 * 4: REM ERASE FUEL INCASE
230 PD = PD + HK*F4 : F4 = U1*F4 : F4 = SF(F4 + F7 + VT): REM NEW CRAFT LOCATION
240 COLOR= 15: REM FALL INTO CRAFT (HANGING) ROUTINE
250 HLIN F4+F7, A7, PD: HLIN F4+F4 * C1, A7, PD + C3: HLIN F4+F7, A7, PD + C2: RETURN : REM DRAW CRAFT
260 REM ASTEROIDS
270 FOR I = C3 TO A5
280 COLOR= I: PLOT A(I+C3),A(I+C2)
290 A(I+C2) = A(I+C2) + S0
300 IF A(I+C2) > A7 THEN GOTO 340
310 COLOR= A(I+C2): PLOT A(I+C3),A(I+C2)
320 NEXT
330 RETURN
340 REM NEW ASTEROID
350 A(I+C2) = INT ( RND (C1) * C3)
360 A(I+C3) = INT ( RND (C1) * 39 + C1)
370 A(I+C3) = INT ( RND (C1) * 14 + C1)
380 IF A(I+C3) > 23 AND A(I+C3) < 17 AND A(I+C2) > 7 THEN A(I+C2) = 7
390 RETURN
400 REM PROGRAM START
410 C0 = 0: C1 = 1: C2 = 2: C3 = 3: REM CONSTANTS
420 GOSUB 800: REM INITIALISE TONE ROUTINE
430 GOSUB 1501: REM INSTRUCTIONS
440 S0 = 3: REM SPEED OF ASTEROIDS
450 A5 = 21: REM NO OF ASTEROIDS
460 DIM A(A5-C3): REM ASTEROID X+Y LOCATION
470 S1 = 5: REM NO OF SPACECRAFT
480 F4 = 4*(F5 + 4): F7 = F5 - INT ( RND (C1) * 3): REM INITIAL SPACECRAFT LOCATION
490 REM SET UP ASTEROID LOCATIONS
500 FOR I = C3 TO A5
510 A(I+C2) = INT ( RND (C1) * 40): REM RANDOM X LOCATION
520 GOSUB 340: REM NEW ASTEROID
530 IF ABS (A(I+C1) - F5 - C1) < 3 AND A(I+C2) > 32 THEN I = I - C1: REM NOT ON SPACECRAFT
540 NEXT
550 POKE - 16368,C0: REM RESET KEYBOARD READ
560 IF F$ = "S" THEN SET F10
570 F$ = "S"
580 HOME
590 POKE - 16304,C0: REM SWITCH TO GRAPHICS
600 POKE - 16302,0: REM FULL OR SCREEN
610 CALL - 1998: REM CLEAR OR SCREEN
620 COLOR= 11: HLIN 19,19 AT C0: HLIN 21,22 AT C0: VLIN C1,C3 AT 18: VLIN C1,C3 AT 22: REM DRAW SPACE-STATION
630 COLOR= 9: VLIN C0,47 AT 10*FU + C0: REM FUEL
640 COLOR= 15: GOSUB 2500: REM DRAW CRAFT
650 REM MAIN LOOP
660 GOSUB 1401: REM MOVE CRAFT
670 GOSUB 1700: REM MOVE ASTEROIDS
680 COLOR= 11: PLOT 18+C3, F5+C3: REM HIT OR SPACE STATION?
690 GOSUB 20: REM CHECK FOR HIT
700 IF PEEK ( - 16384) < 155 THEN SET Y$: HOME : TEXT : END : REM ESCAPE
710 GOTO 660
720 REM DOCK
730 GOSUB 1240
740 PRINT "YOU HAVE DOCKED SUCCESSFULLY - WELL DONE"
750 PRINT "YOUR SUPPLIES WILL BE SENT IMMEDIATELY."
760 FOR I = 1 TO 3000: NEXT I
770 POKE - 16368,C0: REM RESET KEYBOARD READ
780 PRINT : PRINT "ANOTHER GAME"; : GET Y$: PRINT Y$: IF Y$ = "Y" THEN POP : GOTO 470
790 END
```

```
800  REM SET UP TONE ROUTINE
810  FOR I = &H0 TO &H01: READ Q: POKE I,Q: NEXT
820  DATA 173,48,192,136,208,5,206,131,3,240,9,202,208,245,174,110,3,76,113,3,96
830  RETURN
840  REM EXPLOSION
850  FOR I = C1 TO 20
860  COLOR= INT ( RND (C1) * 13 + C1): GOSUB 250
870  POKE 878,50: REM SOUND PITCH
880  POKE 879,10: REM SOUND DURATION
890  CALL &H01: REM PRODUCE SOUND EFFECT
900  NEXT
910  RETURN
920  REM INSTRUCTIONS
930  TEXT : HOME
940  SB = "**********************************************":SA = "*                                            *"
950  VTAB (7):
960  INVERSE : PRINT SA$(6);"A";SPC( 13);"SPACE SLALOM";SPC( 13);"A";SA$(6,6): NORMAL
970  VTAB (20): PRINT  SPC( 4);"BY GEOFF HALT AND STEVE WITHERS"
980  FOR I = 1 TO 3000: NEXT : HOME
990  PRINT "YOU ARE A LONE OBSERVER ON A REMOTE MOON";
1000  PRINT "AND  ARE  SHORT  OF FOOD. THE FAILURE OF";
1010  PRINT "YOUR RADIO MEANS THAT YOU WILL  HAVE  TO";
1020  PRINT "SEND AN SOS IN ONE OF THE REMOTE-CONTROL";
1030  PRINT "RECONNAISSANCE SPACECRAFT THERE  TO  THE";
1040  PRINT "SPACE STATION.  YOU MUST DOCK ACCURATELY";
1050  PRINT "AND AVOID THE ASTEROIDS.  OR  THE  CRAFT";
1060  PRINT "WILL BE DESTROYED.";
1070  PRINT
1080  PRINT "YOU  ONLY  HAVE  FIVE SPACECRAFT AT YOUR";
1090  PRINT "DISPOSAL.  THE RED BAR ON THE LEFT SHOWS";
1100  PRINT "THE  SPACECRAFT'S FUEL LEVEL. IF IT RUNS";
1110  PRINT "OUT OF FUEL IT  WILL  BE  STRANDED.  THE";
1120  PRINT "AREA  IMMEDIATELY  IN FRONT OF THE SPACE";
1130  PRINT "STATION IS PROTECTED BY A  FORCE  FIELD.";
1140  PRINT "AND ASTEROIDS ARE DEFLECTED ROUND IT.";
1150  PRINT
1160  PRINT "USE PADDLE (1) TO  MOVE  THE  SPACECRAFT";
1170  PRINT "FROM SIDE TO SIDE,  AND PRESS THE  PUSH-";
1180  PRINT "BUTTON TO MOVE IT UPWARDS.";
1190  PRINT : PRINT "PRESS  ESC  TO EXIT GAME.";
1200  VTAB (23): PRINT "PRESS THE SPACE BAR TO START";
1210  RETURN : REM TEST FOR KEY PRESS AT LINE 560
1220  REM FAIL
1230  GOSUB 1340
1240  PRINT "YOU'VE RUN OUT OF SPACECRAFT, AND SO YOU";
1250  PRINT "WILL NEVER RECEIVE YOUR SUPPLIES.";
1260  GOTO 1310
1270  GOSUB 1340
1280  PRINT "SPACECRAFT HAS RUN OUT OF FUEL BLOCKING";
1290  PRINT "THE SPACE STATION ENTRANCE. IT WILL NOT";
1300  PRINT "BE REMOVED IN TIME TO SAVE YOU.";
1310  PRINT : PRINT "YOU ARE DOOMED!!"
1320  GOTO 760
1330  REM RETURN TO TEXT
1340  FOR Z = 1 TO 5: PRINT  CHR$ (7);: FOR ZZ = 1 TO 100: NEXT ZZ,Z: TEXT : HOME : RETURN
1350  REM ***********************************************
1360  REM *** SPACE SLALOM BY GEOFF HALT AND STEVE WITHERS ***
1370  REM ***********************************************
1380  REM YOU CAN ALTER THE NUMBER OF SPACECRAFT OR ASTEROIDS
1390  REM BY CHANGING THE VARIABLES AT LINES 450 AND 470.
```

Robot Nim for PET

by Bob Chappell

We include this program, not because Nim is particularly original nor because it is fast (it isn't), but because the graphics are great fun.

```
5 REM**BOB CHAPPELL***14/5/80***
10 DIMA(3,5)
22 PRINT"■■■■■■■■■■■■ROBOT NIM
24 PRINT"■■■DO YOU WANT INSTRUCTIONS (Y/N)";:INPUTA$
26 IFA$="N"THEN70
30 PRINT"■THE GAME IS FOR 2 PLAYERS."
32 PRINT"■THE AIM IS TO LEAVE YOUR OPPONENT WITH
34 PRINT"■ONLY 1 ROBOT LEFT ON THE SCREEN.
36 PRINT"■YOU TAKE TURNS AND ON EACH TURN SAY
37 PRINT"■■WHICH ROW (1-3) AND HOW MANY ROBOTS YOU
39 PRINT"■WISH REMOVED FROM THAT ROW.
41 PRINT"■YOU MAY ONLY REMOVE THE LEADING ROBOT
43 PRINT"■■HEN HE IS THE LAST LEFT IN THE ROW.
45 PRINT"■FOR SOUND EFFECTS,CONNECT AN AMPLIFIER
47 PRINT"■AND SPEAKER TO PINS M(CB2) AND N(GROUND)
49 PRINT"OF THE USER PORT USING A 15-500 RESISTER
60 PRINT"■■PRESS SPACE KEY WHEN READY"
```


Now get OHIO SUPERBOARD II Computer with quick delivery.

ONLY $335*

Ohio Scientific Superboard II

The first complete computer system on a board. Includes keyboard, video interface and audio cassette interface. BK BASIC-in-ROM; 4K RAM. Requires power supply +5V at 3 Amp.

"We heartily recommend Superboard II for the beginner who wants to get into microcomputers with a minimum cost. A real computer with full expandability."
POPULAR ELECTRONICS, MARCH, 1979

"The Superboard II is an excellent choice for the personal computer enthusiast on a budget."
BYTE, MAY, 1979

To Order:
Telephone, write or call.

Freight:
All orders $150 or more are shipped freight prepaid.

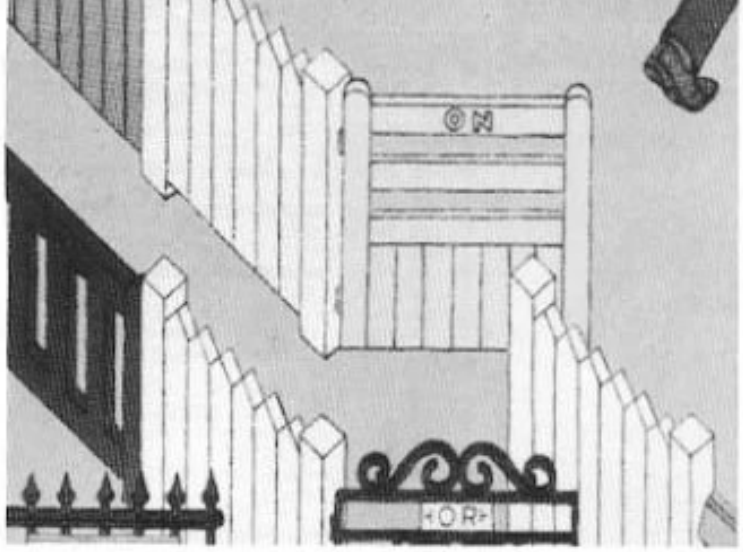
* including $42.00 sales tax

We have moved. Our new address is:

COMPUTERWARE,
305 LATROBE STREET,
MELBOURNE, 3000

CALL 68 4200

```
62 GETA$:IFA$=""THEN62
70 PRINT"■":AA=59467:BB=59466:CC=59464
78 FORK=1TO3:FORJ=0TO5:A(K,J)=0:NEXT:NEXT
80 P=1:T=1:S$=" ":FORJ=1TO6
90 PRINT"■"TAB(T);:GOSUB100
96 T=T+7:NEXT:GOTO120
100 PRINT"■ ■ ■■■■ ■■■■■"S$"■■■■■■  ■■■■■■
```



Next Month

GATEWAYS TO LOGIC
For the first time, a series on the art of teaching others about micros – by Derrick Daines.

BENCHTEST
The HP85 is Hewlett Packard's long awaited entry into the small computer market and has many "plus" features – and some suprising omissions. Has it been worth the wait?

BEST OF BOTH WORLDS
RM Yorston explains the practicalities of adding a second and different family of chip to your micro.

RANDOM WRITINGS RESUMED
Picking up the threads of a previous article, Michael James moves on to deal with some of the uses to which random numbers can be put.

COMPUTER GAMES
Part 4 of the series . . . speeding up the search!

FACE TO FACE
The first part in a series by David Hebditch in which he will be looking at some revealing examples of design incompetence at the human interface.

NEWCOMERS START HERE
A brief guide for the microcomputing novice.

PASCAL
Sue Eisenbach and Chris Saddler discuss the application of Pascal to the storage and manipulation of data within programs.

BOOKFARE
Alvin Toffler's *The Third Wave* reviewed by Malcolm Peltu.

PLUS REGULAR FEATURES
Newsprint, Yankie Doodles, Systems, Buzzwords, In Store, Users Group Index, Programs, Fax and Leisure Lines.

ADVERTISERS INDEX

Anderson Digital Equipment 26
Antelope Engineering 25
BS Microcomp. 26
C.I.S.A. 32
Computerland . . . . . IFC, IBC
Computerware. 54
Cottage Computers 30
Datasoft 52
De Forest Software . . . . . Insert, 6
Hanimex 2
Informative Systems 20
McGills Authorized Newsagency 12
MicroPro Design 16
MS Microsoftware 36, 44
Logic Shop 17
Looky Video. 23
Seahorse Software. 50

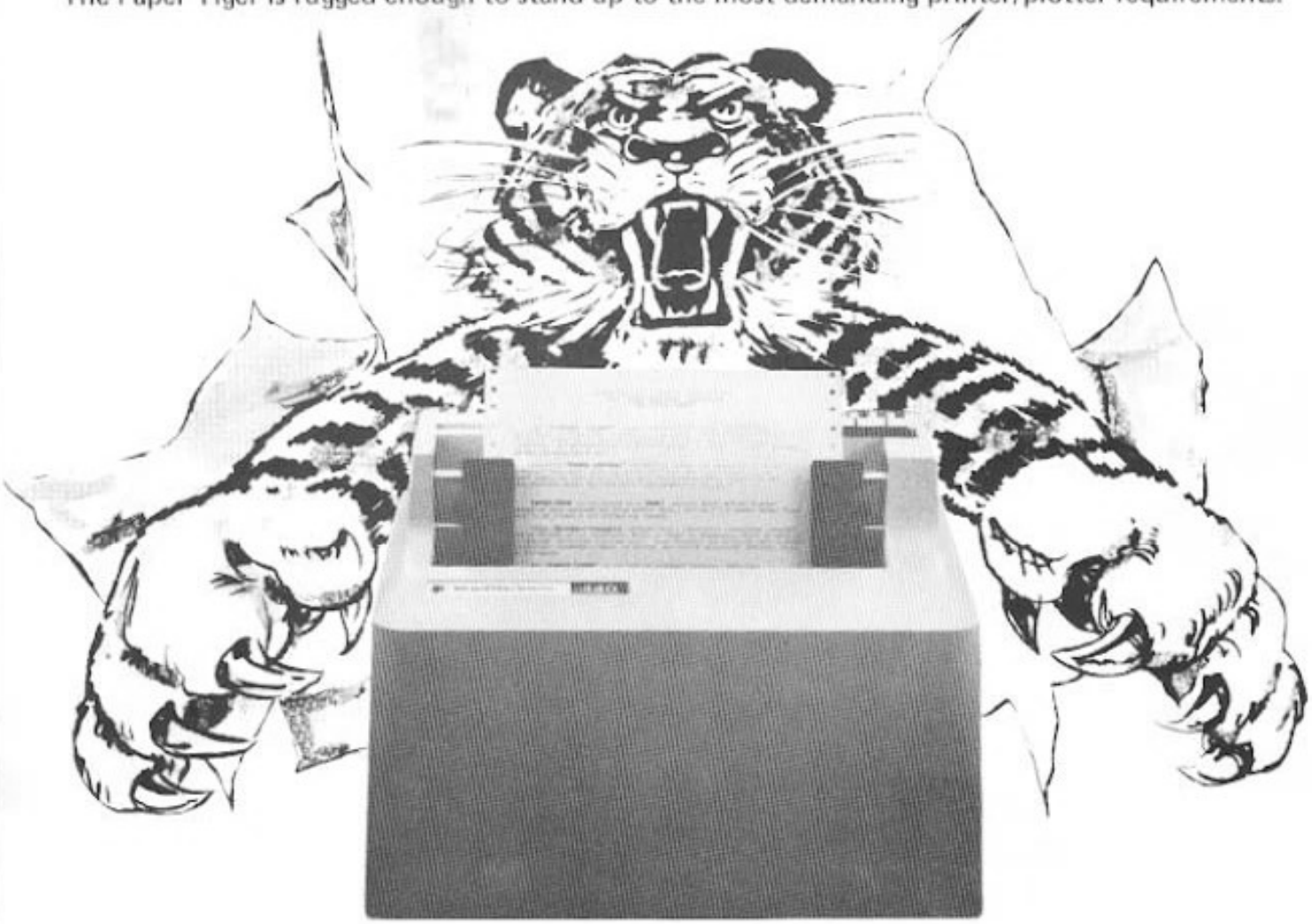
The Paper Tiger IDS440 Impact Printer

Puts more bite into everything you do.

Now available — the Paper Tiger Printer features a graphics option that lets you make the most of your Apple II or other personal computer. And, the Paper Tiger gives you 8 software-selectable character sizes, 80 and 132 column formats, Multi-Part business forms handling, adjustable tractor feed, form control, reliable stepper motor paper drive, serial and parallel interfacing . . . plus lots more.

Don't let our low prices fool you!

The Paper Tiger is rugged enough to stand up to the most demanding printer/plotter requirements.



We also stock:

TEXAS INSTRUMENTS: High speed, high reliability, professional dot matrix, 132 character/line printers.

N.E.C.: Highest standard in professional character printers, for word processing and similar applications.

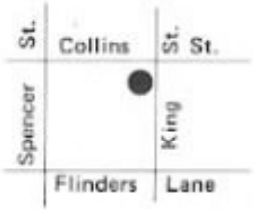
— AUSTRALIA'S LEADERS IN SMALL COMPUTER SYSTEMS —

write now or call us:

ComputerLand®

in Melbourne

Grd. Floor, 555 Collins Street, MELBOURNE VIC. 3000.
Phone (03) 62 6737 (03) 62 5581 Telex AA37007.



The Challenger 4

Whichever way you look at it, no other computer offers so much for so little, and in colour!

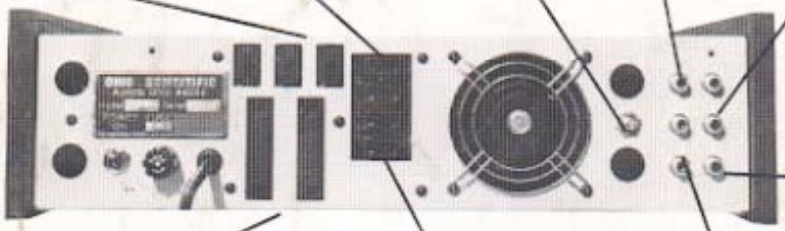
Minifloppy – two can be added.

Full 53 key keyboard which in polled mode can recognise up to 8 simultaneously depressed keys.

RF shielded aluminium case with 2 step baked on enamel finish.

Inside: real time clock and countdown timer 24K memory 6502A processor.

Solid oiled walnut sides.



Control line interfaces, parallel lines for home security, accessory BUS, parallel I/O lines or other TTL signals.

Joystick sockets – add realism to games.

Colour adjustment.

Video Display 64 × 32

Sound outputs, programmable tone generators – 8 bit companding digital to analog convertors.

RF output for control of AC appliances.

Printer RS232 interface.

Numeric key pad slots.

Cassette sockets.

You'd have to go a long way to get better value in a computer. It has execution speed that really separates the computers from the toys. We think the Challenger 4 is way ahead of anything you've seen so far, for a wide variety of uses including business, personal, educational and games, as well as a real-time operating system, word processor and a data base management system.

The Challenger 4 has a 2MHz 6502 processor, and if that's not fast enough we can supply the GT option with the 6502C processor, and 120 nanosecond memory which averages over one million instructions per second.

A real time clock and count down timer, a 64 x 32 display in 16 colours, including 8K memory in the cassette version, 24K for the minifloppy. A BUS structure allows easy plug in of extra memory or many more OHIO boards. The BUS means modularity. If you bought your vintage C2-4 in 1977 we can change the boards at a much lower cost than a new computer.

For the best surprise of all ask our opposition if they can provide all these facilities. When they can't, ask us!

Special offer with this advertisement only — bring it along with you when you visit your dealer and obtain $20 discount off your CHALLENGER 4 purchase.

TCG

TOMORROWS TECHNOLOGY TODAY

OHIO SCIENTIFIC

LOCKWOOD TCG 4Es

For more information and advice call on your local dealer to help you select the best system for your needs

AUSTRALIAN DISTRIBUTOR-TCG, 31 Hume St., Crows Nest, N.S.W. 2065

AUTHORISED AUSTRALIAN AGENTS

NEW SOUTH WALES

Bambach Elect. NEWCASTLE 2-4996
Trevor Burton Pty. Ltd. BALGOWLAH 94 3861
Compuserve Newcastle Pty. Ltd. HAMILTON 61 2579
Dwell Electronics HORNSBY 487 3111
Hi-Fi Gallery TAMWORTH 66 2525
Macelec WOLLONGONG 29 1455
Manly Stat. Suppliers MANLY 977 2316
Micro Visions KINGSFORD 662 4063
MRM Computer Services CROWS NEST 439 2222
J.G. Pearce Systems DOVER HEIGHTS 789 4300
Unique Electronics MERRYLANDS 682 3325

AUSTRALIAN CAPITAL TERRITORY

MES CANBERRA 82 1774

QUEENSLAND

Dialog BRISBANE 221 4898
Johannsen Systems MT. ISA 43 5911

SOUTH AUSTRALIA

Applied Data Control FULLARTON 79 9211
K Tronics ADELAIDE 212 5505

TASMANIA

Aero Electronics HOBART 34 8232
Eastside Computers EAST DEVENPORT 27 8121
J. Walsh & Co. HOBART 34 7511

VICTORIA

Computerware FOOTSCRAY 68 4200
Cottage Computers FITZROY NORTH 481 1975
Data Dimensions SOUTH MELBOURNE 690 1166
Ellison Hawker Education Pty. Ltd. HAMPTON 598 9141

WESTERN AUSTRALIA

Datas Comp. Acc. PERTH 381 3648
Micro Data EAST PERTH 328 1179
Micro Solutions SUBIACO 381 8372